Molekulardynamik und Molekulargraphik von Modellflüssigkeiten

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften
(Dr. rer. nat.)

dem Fachbereich Chemie
der Philipps-Universität Marburg
vorgelegt von
Adrian Gabriel
aus Herborn

Marburg/Lahn 2010

Vom Fachbereich Chemie der Philipps-Universität Marburg als Dissertation angenommen am

Erstgutachter: Prof. Dr. Guido Germano

Zweitgutachter:

Tag der mündlichen Prüfung am

Einleitung

In den letzten Jahrzehnten hat die Bedeutung der Computersimulation in Forschung und Entwicklung stetig zugenommen. Kaum eine Firma, die nicht ihr Produkt zuvor auf dem Computerprüfstand testet, um die großen Kosten realer Tests zu sparen. Pharmafirmen suchen mit Hilfe von ausgeklügelten Simulationsberechnungen nach neuen Wirkstoffen, bevor diese durch Synthese in die reale Welt gebracht und auf tatsächliche Wirksamkeit geprüft werden. So können auch Theorien und Experimente aus z. B. chemischen und physikalischen Zusammenhängen mit Hilfe der Computersimulation getestet werden.

Simulationsberechnungen oder auch Computerexperimente nehmen eine Stellung zwischen Theorie und Experiment ein. Einerseits gibt es in der Theorie stets Problemstellungen, die sich mathematisch mit heutigen Methoden noch nicht lösen lassen, und dies vielleicht auch nie möglich sein wird. Oft kann jedoch die unbekannte analytische Lösung gut mit einem numerischen Verfahren genähert werden. Sollten Theorie und Experiment nicht übereinstimmen, wäre es ohne Computersimulation nicht möglich festzustellen, ob die Theorie oder deren Näherung falsch ist. Vorhersagen der Theorie und andere Fragestellungen können unter Umständen experimentell nicht nachprüfbar sein, sei es nun, dass das Experiment zu kompliziert, zu teuer oder schlicht nicht möglich ist. In allen diesen Fällen kann ein Computerexperiment das Mittel sein, welches zum Erfolg führt und die Lücke zwischen Theorie und Experiment zu schließen in der Lage ist. Computerexperimente dürfen also nicht als Ersatz für reale Experimente betrachtet werden und ebenso nicht für grundsätzliche Theorien. Es ist vielmehr so, dass Simulationsberechnungen eine eigene Klasse bilden. Sie sind ein neues Werkzeug, um Wissenschaft und Technik auf der Suche nach Erkenntnissen ein Stück voranzubringen.

Die vorliegende Arbeit nutzt solche computergestützten Experimente um Theorien aus dem Rahmen der physikalischen Chemie zu überprüfen und zu entwickeln. Kapitel 1 bietet eine theoretische Einleitung in übliche Vorgehensweisen und Algorithmen bei Molekulardynamikberechnungen. In den Kapiteln 2 und 3 werden die gezeigten Methoden angewendet, um Simulationen am Lorentzgas und an Gasen aus harten zwei- und dreidimensionalen Kugeln durchzuführen. Im ersten Fall zielen die Berechnungen auf die Überprüfung einer Fluktuationstheorie für stationäre Zustände von Nichtgleichgewichtssystemen, in deren Rahmen ein asymmetrisches Verhalten großer Fluktuationen unter gewissen Bedingungen vorhergesagt wird. Die Integration der Elektronenbahnen erfolgt

mit dem sogenannten Velocity-Verlet-Algorithmus, welcher im Verlauf von Abschnitt 1.1 über die zeitgesteuerte Molekulardynamik vorgestellt wird. Im zweiten Falle ist das Ziel die Bestimmung von übergeordneten und allgemeingültigen Verteilungsfunktionen für beliebige Teilchenzahlen und Dimensionen, welche im thermodynamischen Grenzfall in die bekannten Verteilungen für Energie, Teilchengeschwindigkeitskomponenten (Maxwell-Boltzmann) und -beträge (Maxwell) übergeht. Die verwendeten Methoden zur Simulation von Fluiden harter Kugeln werden in Abschnitt 1.2 über die ereignisgesteuerte Molekulardynamik vorgestellt.

Die Computersimulation ist noch ein verhältnismäßig neues Feld, und es fehlen an einigen Stellen dringend benötigte Hilfsmittel. Ein Beispiel hierzu ist das Fehlen eines Grafikprogramms, welches in der Lage ist, eine Vielzahl von starren Körpern, wie sie zur „coarse-grained"-Näherung von Molekülen und Molekülteilen eingesetzt werden, effizient darzustellen. In Kapitel 4 wird ein solches Programm vorgestellt, welches von Beginn an im Rahmen dieser Arbeit entwickelt und gepflegt wurde. Das Programm ist „open source" und wird frei im Internet zur Verfügung gestellt.

Inhaltsverzeichnis

1 Molekulardynamik **1**

1.1 Zeitgesteuerte Molekulardynamik 1

1.1.1 Verlet-Algorithmus 5

1.1.2 Leapfrog-Algorithmus 6

1.1.3 Velocity-Verlet-Algorithmus 7

1.1.4 Liouville-Formulierung 8

1.2 Ereignisgesteuerte Molekulardynamik 13

2 Untersuchung großer Fluktuationen im Lorentz-Gas **19**

2.1 Motivation 19

2.2 Aufbau der Simulation 20

2.3 Aufbau und Test der Analysesoftware 27

2.4 Auswertung 32

3 Verteilungen von Energie und Geschwindigkeit in mikrokanonischen Gesamtheiten harter Kugeln **45**

3.1 Motivation 45

3.2 Einleitung 46

3.3 Aufbau der Simulation 48

3.4 Programminterna 52

3.5 Zusammenfassung der Ergebnisse 55

3.6 Auswertung 57

3.6.1 Auffälliges Verhalten bei drei harten Scheiben in Box mit PR 68

3.6.2 Vergleich des Verhaltens von drei harten Scheiben mit PR und vier harten Scheiben in punktsymmetrischer Anordnung mit HW 68

3.6.3 Vergleich des Verhaltens von vier harten Scheiben mit PR und sechs harten Scheiben in punktsymmetrischer Anordnung mit HW 72

3.6.4 Vergleich des Verhaltens von sieben harten Scheiben mit PR und zwölf harten Scheiben in punktsymmetrischer Anordnung mit HW . 77

3.6.5 Das Verhalten zweier harter Scheiben in einer Box mit HW 77

3.6.6 Bemerkungen zum dreidimensionalen Fall 79

3.7 Diskussion 84

4 Molekulargraphik von *coarse-grained*-Flüssigkeiten 85
4.1 Motivation 85
4.2 Programmfeatures 89
4.2.1 Gerenderte und vereinfachte Darstellung 90
4.2.2 Farbkodierung 90
4.2.3 Benutzer-Interface 92
4.2.4 Bildschirmfotos 92
4.2.5 Schnitte 94
4.2.6 Videos 94
4.2.7 Gemische 95
4.2.8 Periodische Randbedingungen 95
4.2.9 Licht- und Farboptionen 97
4.2.10 Renderqualität 98
4.2.11 Remotezugriff auf Dateien 98
4.2.12 Speichern von Optionen 98
4.3 Programminterna 100
4.3.1 Die Programmstruktur 100
4.3.2 Benutzerspezifische Anpassung 100
4.4 Performance 104
4.4.1 Szenengraph gegen direktes Rendern 104
4.4.2 Display Lists gegen Vertex Buffer Objects 106
4.4.3 Detailstufe 107
4.4.4 Occlusion query 108
4.4.5 Backface culling 108
4.4.6 Benchmarkergebnisse 108

Literaturverzeichnis 115

Kapitel 1

Molekulardynamik

1.1 Zeitgesteuerte Molekulardynamik

In der Molekulardynamik (MD) werden Informationen über ein System durch Zeitmittel gewonnen, also durch die direkte Berechnung aller Teilchentrajektorien und damit der Trajektorie des gesamten Systems im Phasenraum. Im Gegensatz dazu steht Monte Carlo (MC), eine Methode, welche dieselbe Informationen aus Ensemblemitteln berechnet. Ist ein System ergodisch, liefert ein Zeitmittel denselben Wert wie ein Ensemblemittel. Eine Übersicht von wichtigen Ensembles bzw. Gesamtheiten ist in Tab. 1.1 zu lesen.

Name	Vorgegebene Variable
mikrokanonisch	N, V, E
isoenthalpisch	N, P, H
kanonisch	N, V, T
isotherm-isobar	N, P, T
großkanonisch	μ, V, T

Tabelle 1.1: Übersicht von wichtigen Gesamtheiten in der statistischen Mechanik. Es bezeichnet N die Teilchenzahl, V das Volumen, E die Energie, P den Druck, H die Enthalpie, T die Temperatur und μ das chemische Potential des betrachteten Systems.

Für viele untersuchte Materialien ist die Annahme sinnvoll, dass die einzelnen Teilchen (Moleküle, Atome), aus denen sie zusammengesetzt sind, sich gemäß der klassischen Mechanik verhalten. Dabei ist klassisch sowohl in dem Sinne nicht quantenmechanischer, als auch nicht relativistischer Natur zu verstehen. Die Relativitätstheorie spielt insofern tatsächlich keine Rolle, dass die auftretenden Geschwindigkeiten klein genug sind, um sie guten Gewissens klassisch betrachten zu können. Die Vernachlässigung der Quantenmechanik hingegen stellt meist eine Vereinfachung dar, um die Systeme, vom zeitlichen Aufwand her gesehen, überhaupt einer Berechnung zugänglich zu machen. Während es in

einigen Bereichen der realistischen Simulation von Materialien, z.B. von Biomakromolekülen in wässriger Lösung, im Prinzip wünschenswert wäre, ab initio rechnen zu können*, sind in Flüssigkristallen quantenmechanische Effekte vernachlässigbar und oft sogar eine atomisch detaillierte Beschreibung überflüssig†. Andere innerhalb dieser Arbeit betrachtete Systeme (Lorentzgas, harte Kugeln) sind modellhafter Natur und per se klassisch definiert, da sie mit Theorieergebnissen aus der klassischen Mechanik verglichen werden sollen.

Bei einem klassischen System N gleichartiger Teilchen der Masse m sind die Newtonschen Bewegungsgleichungen zu lösen,

$$m\frac{d^2\vec{r}_i}{dt^2} = \vec{f}_i\left(\vec{r}\right), \quad i = 1, \ldots, N. \tag{1.1}$$

Dabei bezeichnet $\vec{r} = \{\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_i, \ldots, \vec{r}_N\}$ die Position aller Teilchen und $\vec{f}_i\left(\vec{r}\right)$ die Kraft, die alle anderen Teilchen auf Teilchen i aufgrund der gegenseitigen Abstände $\vec{r}_{ij}$ ausüben. Typischerweise sind die wirkenden Kräfte konservativ, und damit existiert ein Potential $U\left(\vec{r}\right)$, so dass

$$\vec{f}_i\left(\vec{r}\right) = -\frac{\partial}{\partial\vec{r}_i}U\left(\vec{r}\right). \tag{1.2}$$

Dieses wird durch ein effektives Paarpotential

$$U\left(\vec{r}\right) = \sum_{i<j} U_{ij}\left(\vec{r}_{ij}\right), \tag{1.3}$$

welches sich aus der Art der zugrunde liegenden Wechselwirkung ergibt, genähert.[1] Für ungeladene Teilchen kommt häufig das Lennard-Jones-(12,6)-Potential

$$U_{ij}\left(r_{ij}\right) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{1.4}$$

*Bei in Wasser gelösten Biomakromolekülen treten Effekte auf, welche sich ausschließlich mit quantenmechanischen Mitteln adäquat beschreiben lassen. Als Beispiele seien 1. der Protonenaustausch, also Säure-Base-Effekte, und 2. die Bildung von Wasserstoffbrücken genannt. Jedoch ist eine quantenmechanische Berechnung aufgrund der großen Anzahl an Atomen im System derzeit noch unmöglich. Ein typisches Protein besteht aus einigen Tausend Atomen; dazu kommen noch einige Zehntausend Atome Lösungsmittelwasser. Selbst klassisch ist bei der aktuellen Rechnerleistung lediglich eine Zeitskala in der Größenordnung von Nanosekunden berechenbar; jedoch liegen interessante Phänomene, z. B. die Proteinfaltung, oft im Bereich von Millisekunden. Zur Verdeutlichung: Ein einzelner Zeitschritt liegt in der Größenordnung eine Femtosekunde (10^{-15} s). Für eine Nanosekunde (10^{-9} s) müssen also 1 Million Zeitschritte berechnet werden. Da eine Millisekunde (10^{-3} s) aus wiederum 1 Million Nanosekunden besteht, wären insgesamt 10^{12}, also 1 Billion Zeitschritte nötig.

†Wegen der großen Zahl der Atome eines Flüsigkristallmoleküls ($\gtrsim 40$) und der damit verbundenen noch größeren Anzahl an Elektronen ist die quantenmechanische Berechnung eines einzigen Zeitschritts für ein System mit einigen tausend Molekülen sehr aufwändig. Die Zahl der Atome für das gesamte System liegt mindestens in der gleichen Größenordnung wie auch für die zuvor erwähnten Biomoleküle in wässriger Lösung, da übliche Systemgrößen bei mindestens 1 000 Molekülen beginnen, müssen aber auch 100 000 überschreiten können. Es ist offensichtlich, dass solche Systeme nicht quantenmechanisch behandelt werden können, und Verfahren zur Vereinfachung angewandt werden müssen. Im Gegensatz zu den Biomolekülen ist dies aber meist mit kleineren Näherungen möglich.

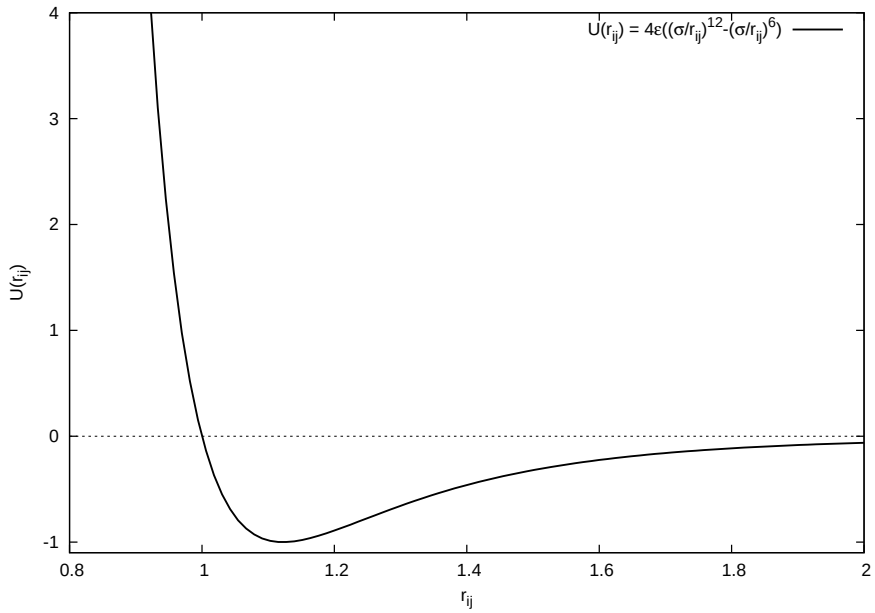


Abbildung 1.1: Diese Abbildung zeigt das Lennard-Jones-(12,6)-Potential, siehe Gleichung (1.4), welches in der Molekularsimulation verwendet wird, um die Wechselwirkung ungeladener nicht kovalent gebundener Atome zu beschreiben. Der Abstand zweier Teilchen ist hier mit r_{ij}, das Potentialminimum mit ϵ und die Nullstelle mit σ bezeichnet. Für die Abbildung wurden die üblicherweise verwendeten Werte $\sigma = 1$ und $\epsilon = 1$ benutzt.

zum Einsatz; siehe Abb. 1.1. Beispielsweise wurde die Simulation flüssigen Argons im Jahr 1964 von Rahman[2] mit dem Lennard-Jones-Potential durchgeführt. Es handelt sich dabei um die erste Computersimulation einer realistischen Flüssigkeit. Einige Jahre zuvor waren bereits idealisierte Flüssigkeiten aus harten Scheiben von Alder und Wainwright simuliert worden.[3] Noch etwas weiter zurück liegt die MC-Simulation harter Kugeln von Metropolis, Rosenbluth und Teller aus dem Jahr 1953/54.[4,5]

Im Allgemeinen werden in der MD Systeme mit einer Teilchenzahl von $N \gg 2$ untersucht. Damit handelt es sich bei Gleichung (1.1) um ein analytisch nicht lösbares Vielteilchenproblem. Aus diesem Grund muss Gleichung (1.1) numerisch integriert werden. Im den folgenden Abschnitten werden einige Algorithmen vorgestellt, die sich zur numerischen Integration der Bewegungsgleichungen in der mikrokanonischen Gesamtheit eignen. Die Arbeitsweise der verschiedenen Algorithmen ist in Abb. 1.2 schematisch dargestellt. Es existieren Erweiterungen, mit denen Thermostaten und Barostaten und somit andere Ensembles als das mikrokanonische simuliert werden können.[6–10]

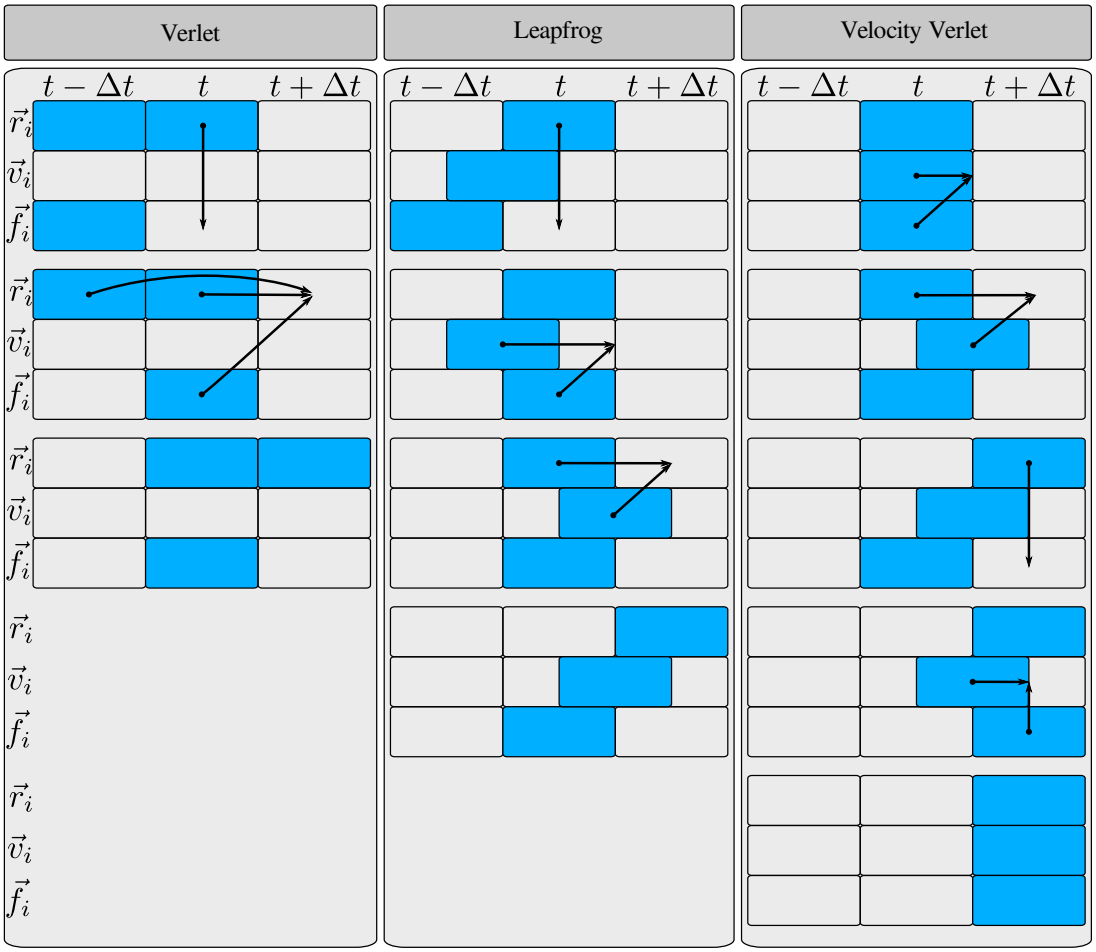


Abbildung 1.2: Die Abbildung zeigt schematisch die Arbeitsweise der verschiedenen Varianten der Verlet-Familie. Zu sehen sind Verlet (links), leapfrog (Mitte) und velocity Verlet (rechts). Dargestellt in den Spalten ist jeweils ein kompletter Rechenzyklus. Die farbig hervorgehobenen Kästchen zeigen die zu diesem Arbeitsschritt bekannten Daten an. Dabei bezeichnen $\vec{r}_i$, $\vec{v}_i$ und $\vec{f}_i$ Orte, Geschwindigkeiten und Kräfte von Teilchen i, sowie $t - \Delta t$, t und $t + \Delta t$ den vergangenen, den aktuellen und den nächsten Zeitschritt.

1.1.1 Verlet-Algorithmus

In einfacher Weise geht dieser Algorithmus aus den Taylorreihenentwicklungen der Teilchenorte zu den Zeitpunkten $t + \Delta t$ und $t - \Delta t$ hervor,[1]

$$\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \dot{\vec{r}}_i(t)\,\Delta t + \frac{\ddot{\vec{r}}_i(t)}{2}\Delta t^2 + \frac{\dddot{\vec{r}}_i}{3!}\Delta t^3 + \mathcal{O}\left(\Delta t^4\right), \tag{1.5}$$

$$\vec{r}_i(t-\Delta t) = \vec{r}_i(t) - \dot{\vec{r}}_i(t)\,\Delta t + \frac{\ddot{\vec{r}}_i(t)}{2}\Delta t^2 - \frac{\dddot{\vec{r}}_i(t)}{3!}\Delta t^3 + \mathcal{O}\left(\Delta t^4\right). \tag{1.6}$$

Hier sind $\dot{\vec{r}}_i(t) = \vec{v}_i(t)$, $\ddot{\vec{r}}_i(t) = \dot{\vec{v}}_i(t) = \vec{a}_i(t)$ und $\dddot{\vec{r}}_i(t) = \dot{\vec{a}}_i(t) = \vec{j}(t)$ die Geschwindigkeit, die Beschleunigung und der Ruck des Teilchens i. Durch Addition der Gleichungen (1.5) und (1.6) ergibt sich

$$\vec{r}_i(t+\Delta t) + \vec{r}_i(t-\Delta t) = 2\vec{r}_i(t) + \vec{a}_i\Delta t^2 + \mathcal{O}\left(\Delta t^4\right), \tag{1.7}$$

$$\vec{r}_i(t+\Delta t) = 2\vec{r}_i(t) + \vec{a}_i\Delta t^2 - \vec{r}_i(t-\Delta t) + \mathcal{O}\left(\Delta t^4\right). \tag{1.8}$$

Aus $d^2\vec{r}_i/dt^2 = \vec{a}_i$ und Gleichung (1.1) folgt unmittelbar

$$\vec{r}_i(t+\Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t-\Delta t) + \frac{1}{m}\vec{f}_i(\vec{r}(t))\,\Delta t^2 + \mathcal{O}\left(\Delta t^4\right). \tag{1.9}$$

Aus Gleichung (1.9) ist ersichtlich, dass die Orte der Teilchen zum Zeitpunkt $t + \Delta t$ mit einem Fehler der Ordnung Δt^4 von den Orten zu den Zeiten $t - \Delta t$ und t sowie den Kräften zur Zeit t abhängen. Die Kräfte hängen dabei ebenfalls lediglich von den Orten der Teilchen zur Zeit t über $\vec{a}_i = \frac{1}{m}\vec{f}_i = -\frac{1}{m}\frac{\partial}{\partial\vec{r}_i}U(\vec{r})$ ab. An dieser Stelle sei bemerkt, dass es sich bei der Fehlerangabe um die Kurzzeitstabilität handelt. Die Langzeitstabilität der Mitglieder der Verlet-Familie ist $\mathcal{O}(\Delta t^2)$.

Die Geschwindigkeiten tauchen in dieser Darstellung nicht mehr explizit auf. Dieser Umstand ist unpraktisch, da die genaue Kenntnis aller Teilchengeschwindigkeiten zu jedem Zeitschritt erforderlich ist, um eine Reihe wichtiger Größen des Systems bestimmen zu können. Die Prüfung der Energieerhaltung beispielsweise ist ein notwendiger Konsistenztest bei MD-Simulationen und verlangt unter Anderem die Berechnung der gesamten kinetischen Energie der Teilchen im System. Dies ist aber nur bei Kenntnis der Teilchengeschwindigkeiten möglich.

Ein Ausdruck für die instantanen Geschwindigkeiten ergibt sich durch Differenzbildung der Gleichungen (1.5) und (1.6),

$$\vec{r}_i(t+\Delta t) - \vec{r}_i(t-\Delta t) = 2\vec{v}_i(t) + \mathcal{O}\left(\Delta t^3\right), \tag{1.10}$$

$$\vec{v}_i(t) = \frac{\vec{r}_i(t+\Delta t) - \vec{r}_i(t-\Delta t)}{2\Delta t} + \mathcal{O}\left(\Delta t^2\right). \tag{1.11}$$

Es zeigt sich mit Gleichung (1.11), dass die instantanen Geschwindigkeiten nur mit einem

relativ hohen Fehler der Ordnung Δt^2 bestimmt werden können. Des weiteren ist zu ihrer Bestimmung zum Zeitpunkt t die Kenntnis der Positionen zur Zeit $t + \Delta t$ nötig. Die Geschwindigkeiten können also nicht zum aktuellen, sondern nur zum vorherigen Schritt bestimmt werden.

Eine weitere Schwierigkeit stellt der erste Zeitschritt dar, da zu Beginn lediglich die Positionen zur Zeit t, nicht aber diejenigen zur Zeit $t - \Delta t$, bekannt sind. Abhilfe schafft als erster Integrationsschritt

$$\vec{r}_i(\Delta t) \approx \vec{r}_i(0) + \vec{v}_i(0)\,\Delta t + \frac{1}{2}\vec{a}_i \Delta t^2 + \mathcal{O}\left(\Delta t^3\right). \tag{1.12}$$

Die Tatsache, dass dieser erste Schritt mit einem Fehler der Ordnung Δt^3 behaftet ist, fällt bei der normalerweise sehr großen Anzahl ($\gtrsim 10^6$) an Simulationsschritten nicht weiter ins Gewicht.

1.1.2 Leapfrog-Algorithmus

Eine algebraisch äquivalente, jedoch auf Rechnern mit endlicher Darstellung von reellen Zahlen numerisch stabilere Variante des Verlet-Algorithmus ist unter dem Namen leapfrog[‡] bekannt und kann wie folgt hergeleitet werden.[1] Sie wurde Anfang der 1970er Jahre aus Simulationen in der Astrophysik übernommen.

Definiert werden die Geschwindigkeiten zu einem halben Zeitschritt $\Delta t/2$ vor und nach der aktuellen Zeit t:

$$\vec{v}_i\left(t + \frac{\Delta t}{2}\right) = \frac{\vec{r}_i(t + \Delta t) - \vec{r}_i(t)}{\Delta t}, \tag{1.13}$$

$$\vec{v}_i\left(t - \frac{\Delta t}{2}\right) = \frac{-\vec{r}_i(t - \Delta t) + \vec{r}_i(t)}{\Delta t}. \tag{1.14}$$

Nach Umstellung von Gleichungen (1.13), (1.14) und (1.8) zu

$$\vec{r}_i(t + \Delta t) - \vec{r}_i(t) = \vec{v}_i\left(t + \frac{\Delta t}{2}\right)\Delta t, \tag{1.15}$$

$$-\vec{r}_i(t - \Delta t) + \vec{r}_i(t) = \vec{v}_i\left(t - \frac{\Delta t}{2}\right)\Delta t, \tag{1.16}$$

$$\vec{r}_i(t + \Delta t) - \vec{r}_i(t) = \vec{r}_i(t) - \vec{r}_i(t - \Delta t) + \vec{a}_i \Delta t^2 \tag{1.17}$$

und anschließendem Einsetzen von Gleichungen (1.15) und (1.16) in (1.17) ergibt sich mit $\vec{a}_i = \frac{1}{m}\vec{f}_i$

$$\vec{v}_i\left(t + \frac{\Delta t}{2}\right) = \vec{v}_i\left(t - \frac{\Delta t}{2}\right) + \frac{1}{m}\vec{f}_i \Delta t. \tag{1.18}$$

[‡]Leapfrog ist das englische Wort für das Kinderspiel Bockspringen. Die Namensgebung kann durch einen Blick auf Abb. 1.2 verstanden werden.

Leider sind in dieser Variante zwar die Geschwindigkeiten explizit vorhanden, jedoch noch immer nicht zur aktuellen Zeit t. Es ist wiederum eine Abschätzung der aktuellen Geschwindigkeiten notwendig mit

$$\vec{v}_i(t) = \frac{1}{2}\left[\vec{v}_i\left(t - \frac{\Delta t}{2}\right) + \vec{v}_i\left(t + \frac{\Delta t}{2}\right)\right]. \tag{1.19}$$

1.1.3 Velocity-Verlet-Algorithmus

Der velocity Verlet ist eine weitere algebraisch äquivalente Variante der Verlet- und leapfrog-Algorithmen und vereint die numerische Stabilität von leapfrog mit einer sinnvolleren Behandlung der Geschwindigkeiten.[1,11]

Der Algorithmus hat die Form

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \vec{v}_i(t)\,\Delta t + \frac{1}{2m}\vec{f}(t)\,\Delta t^2, \tag{1.20}$$

$$\vec{v}_i(t + \Delta t) = \vec{v}_i(t) + \frac{1}{2m}\left[\vec{f}_i(t) + \vec{f}_i(t + \Delta t)\right]\Delta t \tag{1.21}$$

und wird üblicherweise wie folgt implementiert:

1. Bestimmung der aktuellen Kräfte $\vec{f}_i(t)$.

2. Bestimmung der Geschwindigkeiten zum halben Zeitschritt,

$$\vec{v}_i\left(t + \frac{\Delta t}{2}\right) = \vec{v}_i(t) + \frac{1}{2m}\vec{f}_i(t)\,\Delta t. \tag{1.22}$$

3. Bestimmung der Orte zum vollen Zeitschritt,

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \vec{v}_i\left(t + \frac{\Delta t}{2}\right)\Delta t. \tag{1.23}$$

4. Bestimmung der Kräfte zum vollen Zeitschritt,

$$\vec{f}_i(t + \Delta t) = -\frac{\partial}{\partial \vec{r}_i} U\left(\vec{r}_i(t + \Delta t)\right). \tag{1.24}$$

5. Bestimmung der Geschwindigkeiten zum vollen Zeitschritt,

$$\vec{v}_i(t + \Delta t) = \vec{v}_i\left(t + \frac{\Delta t}{2}\right) + \frac{1}{2m}\vec{f}_i(t + \Delta t)\,\Delta t. \tag{1.25}$$

6. Weiter bei 2.

Es ist zu erkennen, dass dieser Algorithmus die Kenntnis sowohl der Positionen, als auch insbesondere der Geschwindigkeiten aller Teilchen zu jedem vollen Zeitschritt gewährleistet. Außerdem ist dieser Algorithmus selbststartend.

1.1.4 Liouville-Formulierung

Eine formale Herleitung des velocity-Verlet-Algorithmus auf Basis der Liouville-Formulierung der klassischen Mechanik geht auf Tuckerman et al. zurück.[12] Sie soll nun im Folgenden näher betrachtet werden, um ein mathematisches Verständnis der physikalischen Eigenschaften zu fördern. Es wird zunächst von einer eindimensionalen Bewegung eines einzelnen Teilchens ausgegangen, die durch die Hamilton-Funktion

$$\mathcal{H}\left(p\left(t\right), q\left(t\right)\right) = \frac{p^2}{2m} + U\left(q\right) \tag{1.26}$$

charakterisiert werden kann. Dabei sind $p\left(t\right)$ und $q\left(t\right)$ die generalisierten, kanonisch konjugierten Impuls- und Ortskoordinaten, m die Teilchenmasse und $U\left(q\right)$ das Potential. Sie werden zusammengefasst als Phasenraumpunkt $\vec{\Gamma}\left(t\right) = \left(p\left(t\right), q\left(t\right)\right)$ bezeichnet. Die Hamilton-Funktion stellt somit die Gesamtenergie des Systems in einem bestimmten Zeitpunkt $\vec{\Gamma}\left(t\right) = \left(p\left(t\right), q\left(t\right)\right)$ im Phasenraum dar. In diesem Fall ist der Phasenraum zweidimensional und kann bildlich dargestellt werden. Abb. 1.3 zeigt eine mögliche Trajektorie. Im Allgemeinen ist die Dimensionalität des Phasenraumes jedoch so groß ($6N$ mit $N =$ Teilchenzahl), dass er sich einer übersichtlichen Darstellung entzieht.

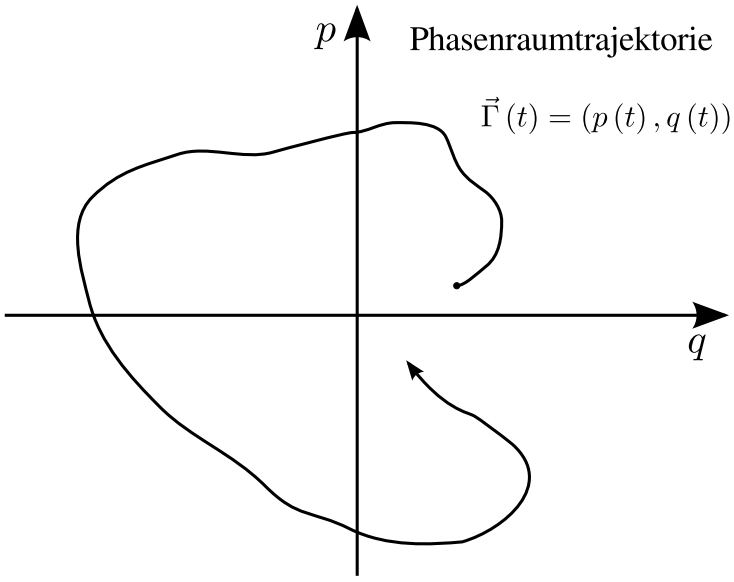


Abbildung 1.3: Diese Abbildung zeigt eine mögliche, beliebige Trajektorie im Phasenraum eines einzelnen Teilchens, welches sich lediglich in einer Dimension bewegen kann. Da der Phasenraum durch die Orts- und Impulskoordinaten aller Teilchen eines Systems aufgespannt wird, ergibt sich in diesem einfachen Fall eine zweidimensionale Ebene.

Definieren wir $\vec{p} = \left(\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_i, \ldots \vec{p}_N\right)$, $\vec{q} = \left(\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_i, \ldots \vec{q}_N\right)$ und $\vec{\Gamma} = \left(\vec{p}, \vec{q}\right)$, ergibt sich die Hamilton-Funktion für ein System mit N Teilchen zu

$$\mathcal{H}\left(\vec{p}, \vec{q}\right) = \frac{p^2}{2m} + U\left(\vec{q}\right), \tag{1.27}$$

Zu integrieren sind die hamiltonschen bzw. kanonischen Bewegungsgleichungen. Es handelt sich bei diesen um die folgenden $6N$ gekoppelten Differentialgleichungen erster Ordnung:

$$\dot{\vec{q}}_i = \frac{\partial \mathcal{H}}{\partial \vec{p}_i}, \tag{1.28}$$

$$\dot{\vec{p}}_i = -\frac{\partial \mathcal{H}}{\partial \vec{q}_i}. \tag{1.29}$$

Diese Gleichungen können auch folgendermaßen dargestellt werden:

$$\frac{\mathrm{d}\vec{\Gamma}}{\mathrm{d}t} = \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{1} & \mathbf{0} \end{pmatrix} \frac{\partial \mathcal{H}}{\partial \vec{\Gamma}}. \tag{1.30}$$

Dabei ist $\mathbf{1}$ eine $3N-$dimensionale Einheitsmatrix und $\mathbf{0}$ eine $3N \times 3N-$dimensionale Nullmatrix. Die daraus zusammengesetzte Matrix ist die sogenannte symplektische Matrix. Gleichung (1.30) bringt gegenüber den Gleichungen (1.28) und (1.29) zwei Vorteile: Erstens handelt es sich nur noch um eine einzige Gleichung und zweitens ist sie frei von Indizes.

Die zeitliche Entwicklung einer beliebigen Funktion der Phasenraumkoordinaten wird beschrieben durch

$$\begin{aligned} \frac{\mathrm{d}\vec{f}\,(\vec{p},\vec{q})}{\mathrm{d}t} &= \frac{\partial \vec{f}}{\partial \vec{q}} \cdot \frac{\mathrm{d}\vec{q}}{\mathrm{d}t} + \frac{\partial \vec{f}}{\partial \vec{p}} \cdot \frac{\mathrm{d}\vec{p}}{\mathrm{d}t} \\ &= \frac{\partial \vec{f}}{\partial \vec{q}} \cdot \frac{\partial \mathcal{H}}{\partial \vec{p}} - \frac{\partial \vec{f}}{\partial \vec{p}} \cdot \frac{\partial \mathcal{H}}{\partial \vec{q}} \\ &=: \left\{ \vec{f}, \mathcal{H} \right\}. \end{aligned} \tag{1.31}$$

Der Ausdruck auf der letzten Zeile heißt Poisson-Klammer. Aus Gleichung (1.31) kann der Liouville-Operator L definiert werden,

$$\begin{aligned} iL &= \frac{\mathrm{d}\vec{q}}{\mathrm{d}t} \cdot \frac{\partial}{\partial \vec{q}} + \frac{\mathrm{d}\vec{p}}{\mathrm{d}t} \cdot \frac{\partial}{\partial \vec{p}} \\ &= \frac{\partial \mathcal{H}}{\partial \vec{p}} \cdot \frac{\partial}{\partial \vec{q}} - \frac{\partial \mathcal{H}}{\partial \vec{q}} \cdot \frac{\partial}{\partial \vec{p}} \\ &= \{\cdot, \mathcal{H}\}. \end{aligned} \tag{1.32}$$

Somit kann Gleichung (1.31) verkürzt in der Form

$$\frac{\mathrm{d}\vec{f}}{\mathrm{d}t} = iL\vec{f} \tag{1.33}$$

geschrieben werden. Die formale Lösung von Gleichung (1.33) lautet

$$\begin{aligned} \vec{f}\left(\vec{p}(t)\,,\vec{q}(t)\right) &= e^{iLt}\vec{f}\left(\vec{p}(0)\,,\vec{q}(0)\right) && (1.34)\\ &= e^{i(L_p+L_q)t}\vec{f}\left(\vec{p}(0)\,,\vec{q}(0)\right), && (1.35)\end{aligned}$$

wobei der Liouville-Operator in zwei Teile

$$iL \equiv iL_p + iL_q \tag{1.36}$$

mit

$$\begin{aligned} iL_p &= \frac{\mathrm{d}\vec{p}}{\mathrm{d}t}\cdot\frac{\partial}{\partial\vec{p}} = \vec{F}\cdot\frac{\partial}{\partial\vec{p}}, && (1.37)\\ iL_q &= \frac{\mathrm{d}\vec{q}}{\mathrm{d}t}\cdot\frac{\partial}{\partial\vec{q}} = \frac{\vec{p}}{m}\cdot\frac{\partial}{\partial\vec{q}} && (1.38)\end{aligned}$$

getrennt wurde.

Die Operatoren iL_p und iL_q vertauschen nicht, das heißt ihr Kommutator

$$[iL_p, iL_q] \equiv iL_p - iL_q = \frac{\mathrm{d}\vec{p}}{\mathrm{d}t}\cdot\frac{\partial}{\partial\vec{p}} - \frac{\mathrm{d}\vec{q}}{\mathrm{d}t}\cdot\frac{\partial}{\partial\vec{q}} \tag{1.39}$$

verschwindet im Allgemeinen nicht.

Die Exponentialfunktion der Summe zweier nicht kommutierender Operatoren A und B kann nicht als das Produkt zweier Exponentialfunktionen geschrieben werden. Das heißt es gilt

$$e^{A+B} \neq e^A e^B \Leftrightarrow [A,B] \neq 0. \tag{1.40}$$

Ein Vergleich der Taylorentwicklungen bis zur zweiten Ordnung beider Seiten von Gleichung (1.40) zeigt dies:

$$\begin{aligned} e^{(A+B)h} &= 1 + (A+B)\,h + \frac{1}{2}\left(A+B\right)^2 h^2 + \mathcal{O}\left(h^3\right)\\ &= 1 + (A+B)\,h + \frac{1}{2}\left(A^2 + AB + BA + B^2\right)h^2 + \mathcal{O}\left(h^3\right). && (1.41)\end{aligned}$$

$$\begin{aligned} e^{Ah}e^{Bh} &= \left(1 + Ah + \frac{1}{2}A^2h^2 + \mathcal{O}\left(h^3\right)\right)\left(1 + Bh + \frac{1}{2}B^2h^2 + \mathcal{O}\left(h^3\right)\right)\\ &= 1 + Ah + Bh + \frac{1}{2}A^2h^2 + \frac{1}{2}B^2h^2 + ABh^2 + \mathcal{O}\left(h^3\right)\\ &= 1 + (A+B)\,h + \frac{1}{2}\left(A^2 + 2AB + B^2\right)h^2 + \mathcal{O}\left(h^3\right). && (1.42)\end{aligned}$$

wobei $h \in \mathbb{R}$ eine beliebige reelle Zahl darstellen soll, um eine Taylorentwicklung um Null zu ermöglichen. Ein Vergleich der Gleichungen (1.41) und (1.42) zeigt, dass die Identität

tatsächlich nur gegeben ist, wenn die Operatoren A und B kommutieren.

Um zu zeigen, dass im allgemeinen der Kommutator $[iL_p, iL_q] \neq 0$ ist, soll ein einfaches Gegenbeispiel herangezogen werden: Könnte davon ausgegangen werden, dass der Kommutator für alle Fälle verschwände, dann müsste dies auch für den des eindimensionalen harmonischen Oszillators gelten. Die auf das oszillierende Teilchen wirkende Kraft

$$F\left(q\left(t\right)\right) = -kq = -m\omega^2 q = -\frac{\mathrm{d}}{\mathrm{d}q}U\left(q\right) \tag{1.43}$$

ergibt sich aus dem Potential

$$U\left(q\left(t\right)\right) = \frac{m\omega^2}{2}q^2 \tag{1.44}$$

wobei k die Federkonstante und $\omega = \sqrt{k/m}$ die Kreisfrequenz darstellen. Die Hamilton-Funktion des harmonischen Oszillators lautet somit

$$\mathcal{H}\left(p\left(t\right), q\left(t\right)\right) = \frac{1}{2m}p^2 + \frac{m\omega^2}{2}q^2. \tag{1.45}$$

Die Bewegungsgleichungen ergeben sich zu

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial\mathcal{H}}{\partial p} = \frac{1}{m}p, \tag{1.46}$$

$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial\mathcal{H}}{\partial q} = -m\omega^2 q. \tag{1.47}$$

Für dieses Beispielsystem stellen wir jetzt den Kommutator auf,

$$[iL_p, iL_q] = \frac{\mathrm{d}p}{\mathrm{d}t}\frac{\partial}{\partial p} - \frac{\mathrm{d}q}{\mathrm{d}t}\frac{\partial}{\partial q} = F\frac{\partial}{\partial p} - \frac{p}{m}\frac{\partial}{\partial q} = -m\omega^2 q\frac{\partial}{\partial p} - \frac{p}{m}\frac{\partial}{\partial q}, \tag{1.48}$$

und betrachten die Funktion der Phasenraumpunkte

$$\vec{f}\left(p\left(t\right), q\left(t\right)\right) = \begin{pmatrix} \frac{\partial\mathcal{H}}{\partial p} \\ -\frac{\partial\mathcal{H}}{\partial q} \end{pmatrix} = \begin{pmatrix} p/m \\ -m\omega^2 q \end{pmatrix}. \tag{1.49}$$

Wird nun der Kommutator aus Gleichung (1.48) auf Gleichung (1.49) angewendet,

$$[iL_p, iL_q]\,\vec{f}\left(p\left(t\right), q\left(t\right)\right) = \left(-m\omega^2 q\frac{\partial}{\partial p} - \frac{p}{m}\frac{\partial}{\partial q}\right)\begin{pmatrix} p/m \\ -m\omega^2 q \end{pmatrix} = -\omega^2\begin{pmatrix} q \\ p \end{pmatrix}, \tag{1.50}$$

lässt sich sofort

$$[iL_p, iL_q] \neq 0 \tag{1.51}$$

ablesen. Damit ist gezeigt, dass iL_p und iL_q im Allgemeinen nicht vertauschen; analog zu Gleichung (1.40) gilt demnach

$$e^{iL_p + iL_q} \neq e^{iL_p}e^{iL_q}. \tag{1.52}$$

Um dennoch mit dem Propagator sinnvoll arbeiten zu können, wenden wir das Trotter-Theorem[13] an. Dieses lautet

$$e^{A+B} = \lim_{h\to 0}\left(e^{\frac{Ah}{2t}}e^{\frac{Bh}{t}}e^{\frac{Ah}{2t}}\right)^{\frac{t}{h}}. \tag{1.53}$$

Schreiben wir $A = iL_p t$, $B = iL_q t$ und interpretieren h als beliebig kleinen Zeitschritt Δt, erhalten wir mit Gleichung (1.35)

$$\vec{f}\left(\vec{p}\left(t\right),\vec{q}\left(t\right)\right) = \lim_{\Delta t\to 0}\left(e^{\frac{iL_p\Delta t}{2}}e^{iL_q\Delta t}e^{\frac{iL_p\Delta t}{2}}\right)^{\frac{t}{\Delta t}}\vec{f}\left(\vec{p}\left(0\right),\vec{q}\left(0\right)\right). \tag{1.54}$$

Gleichung (1.54) ist eine Vorschrift zur Propagation des Systems vom Zeitpunkt 0 hin zum Zeitpunkt t. Die gesamte Propagation wird in $n = t/\Delta t$ einzelnen Schritten der Länge Δt vollzogen. Jeder einzelne Schritt besteht aus der einmaligen Anwendung des sich aus dem Trotter-Theorem ergebenden ein-Schritt-Operators

$$U\left(\Delta t\right) = e^{\frac{iL_p\Delta t}{2}}e^{iL_q\Delta t}e^{\frac{iL_p\Delta t}{2}}. \tag{1.55}$$

Die Wirkmechanik dieses ein-Schritt-Propagators $U\left(\Delta t\right)$ soll im Folgenden geklärt werden. Mit den Definitionen (1.37) und (1.38) der Operatoren iL_p und iL_q besteht ein Propagationsschritt aus der dreimaligen Anwendung eines Operators der Form $e^{k\frac{\partial}{\partial x}}$ auf eine Funktion $\Phi(x)$. Eine Taylorentwicklung der Exponentialfunktion verschafft hier Klarheit,

$$\begin{aligned} e^{k\frac{\partial}{\partial x}}\Phi\left(x\right) &= \sum_{m=0}^{\infty}\frac{k^m}{m!}\frac{\partial^m\Phi\left(x\right)}{\partial x^m} \\ &= \Phi\left(x+k\right). \end{aligned} \tag{1.56}$$

Im Speziellen gilt auch, dass der Operator $e^{k\frac{\partial}{\partial x}}$ bei Anwendung auf eine von x unabhängige Funktion diese reproduziert,

$$\begin{aligned} e^{k\frac{\partial}{\partial x}}\Phi\left(y\right) &= \sum_{m=0}^{\infty}\frac{k^m}{m!}\frac{\partial^m\Phi\left(y\right)}{\partial x^m} \\ &= \Phi\left(y\right). \end{aligned} \tag{1.57}$$

Mit diesen Erkenntnissen ergibt sich zusammen mit den Gleichungen (1.55), (1.37) und (1.38) ohne weiteres Zutun der velocity-Verlet-Algorithmus: Bei der Propagation um einen Zeitschritt werden zu Beginn die Impulse um einen halben Schritt propagiert,

$$\begin{aligned} e^{\frac{iL_p\Delta t}{2}}\left(\vec{p}\left(t\right),\vec{q}\left(t\right)\right) &= \left(\vec{p}\left(t\right)+\frac{\Delta t}{2}\vec{F}\left(t\right),\vec{q}\left(t\right)\right) \\ &= \left(\vec{p}\left(t+\frac{\Delta t}{2}\right),\vec{q}\left(t\right)\right). \end{aligned} \tag{1.58}$$

Anschließend werden die Orte um einen ganzen Zeitschritt propagiert,

$$\begin{aligned} e^{iL_q\Delta t}\left(\vec{p}\left(t+\frac{\Delta t}{2}\right),\vec{q}(t)\right) &= \left(\vec{p}\left(t+\frac{\Delta t}{2}\right),\vec{q}(t)+\Delta t\frac{\vec{p}}{m}\left(t+\frac{\Delta t}{2}\right)\right) \\ &= \left(\vec{p}\left(t+\frac{\Delta t}{2}\right),\vec{q}(t+\Delta t)\right), \end{aligned} \tag{1.59}$$

wobei die Impulse zum halben Zeitschritt zugrunde liegen. Nach der Neuberechnung der Kräfte $\vec{F}(t+\Delta t)$, da diese von den veränderten Orten $\vec{q}(t+\Delta t)$ abhängen, werden zuletzt die Impulse ein weiteres Mal um einen halben Zeitschritt propagiert,

$$\begin{aligned} e^{\frac{iL_p\Delta t}{2}}\left(\vec{p}\left(t+\frac{\Delta t}{2}\right),\vec{q}(t+\Delta t)\right) &= \left(\vec{p}\left(t+\frac{\Delta t}{2}\right)+\frac{\Delta t}{2}\vec{F}(t+\Delta t),\vec{q}(t+\Delta t)\right) \\ &= (\vec{p}(t+\Delta t),\vec{q}(t+\Delta t)). \end{aligned} \tag{1.60}$$

Damit ist die Propagation des Systems um einen Zeitschritt der Länge Δt abgeschlossen.

Bei geeigneter Implementierung der Vektorklassen und Operatoren nimmt der Algorithmus folgende, sehr übersichtliche Form in C++-Programmcode an:

```
// Velocity-Verlet-Algorithmus in C++
vector_velocities += vector_forces*timestep/2;
vector_positions  += vector_velocities*timestep;
vector_forces.compute();
vector_velocities += vector_forces*timestep/2;
```

Die drei Zeilen mit den += Operatoren entsprechen genau der Anwendung der drei ein-Schritt-Propagatoren, bzw. „advancement"-Operatoren $e^{iL_p\Delta t/2}$, $e^{iL_q\Delta t}$ und $e^{iL_p\Delta t/2}$ auf die Phasenraumkoordinate $\vec{\Gamma}(t)=(\vec{p}(t),\vec{q}(t))$, siehe Gleichungen (1.58-1.60).

1.2 Ereignisgesteuerte Molekulardynamik

In der Molekulardynamik gibt es zwei verschiedene Ansätze zur Bestimmung der Teilchentrajektorien. Im Falle kontinuierlicher zwischenmolekularer Potentiale (z.B. Lennard-Jones, siehe Abschnitt 1.1) müssen die Bewegungsgleichungen mit einem zugrunde gelegten konstanten Zeitschritt numerisch integriert werden. Dieser Ansatz wird als zeit-(schritt)gesteuert bezeichnet und wurde in Abschnitt 1.1 erläutert. Dabei bestimmt die Größe der in dem speziellen System wirkenden Kräfte den Zeitschritt. Um die Energieerhaltung nicht zu verletzen, ist bei umso größeren zwischenmolekularen Kräften ein umso kleinerer Zeitschritt erforderlich.

Im Falle harter, also nicht kontinuierlicher, Stufenpotentiale findet die gesamte Wechselwirkung ausschließlich zum Zeitpunkt eines Zusammenstoßes zweier Moleküle statt. Zwischen zweien solcher Stöße bewegen sich die Teilchen hingegen kräftefrei, wenn von

der Existenz eines globalen äußeren Feldes abgesehen wird. In diesem Szenario ist es numerisch stabiler und weniger rechenaufwändig, wenn die Propagation des Systems nicht mit einem zuvor definierten Zeitschritt, sondern von Stoß zu Stoß erfolgt. Einen solchen Ansatz bezeichnet man als ereignisgesteuert. Stöße der Teilchen untereinander sind unter Umständen nicht die einzigen Ereignisse, zu welchen das System propagiert und neu berechnet werden muss. Im Falle periodischer Randbedingungen und bei der Unterteilung der begrenzenden Simulationsbox in Unterzellen[§] kommt ein weiteres Ereignis hinzu, nämlich der Übertritt eines Teilchens von einer Zelle in eine andere. Es sind jedoch auch noch eine ganze Reihe weiterer Ereignisse möglich, z.B. Wandstöße im Falle harter Wände und diverse Zeitpunkte, zu denen bestimmte Messwerte berechnet oder Dateien geschrieben werden sollen. Schlussendlich wird das System mit diesem Ansatz von Ereignis zu Ereignis propagiert. Auf dem Weg zwischen zwei Ereignissen werden die Teilchengeschwindigkeiten i.A. nicht beeinflusst.

Im Folgenden sollen die notwendigen Berechnungen für die Detektion von Stößen in einem System harter Kugeln näher erläutert werden.[1,3] Wir betrachten zwei Teilchen mit Durchmesser σ, welche sich zur Zeit t an den Orten $\vec{r}_i$ und $\vec{r}_j$ befinden und die Geschwindigkeiten $\vec{v}_i$ und $\vec{v}_j$ besitzen. Sei weiterhin der Abstandsvektor definiert als $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$, sowie die Relativgeschwindigkeit als $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$, so ist im Falle eines Stoßes zur Zeit $t + t_{ij}$ folgende Gleichung erfüllt:

$$|\vec{r}_{ij}\,(t + t_{ij})| = |\vec{r}_{ij}\,(t) + \vec{v}_{ij} t_{ij}| = \sigma. \tag{1.61}$$

Mit der Definition des Vektorbetrages und des Skalarproduktes $b_{ij} = \vec{r}_{ij} \cdot \vec{v}_{ij}$ ergibt sich eine quadratische Gleichung in t_{ij},

$$v_{ij}^2 t_{ij}^2 + 2 b_{ij} t_{ij} + r_{ij}^2 - \sigma^2 = 0. \tag{1.62}$$

Die vollständige Lösung dieser Gleichung ergibt sich unmittelbar zu

$$t_{ij}^{1,2} = \frac{-b_{ij} \pm \sqrt{b_{ij}^2 - v_{ij}^2 \left(r_{ij}^2 - \sigma^2\right)}}{v_{ij}^2}. \tag{1.63}$$

Diese Lösung soll nun näher untersucht und in einen physikalischen Sinnzusammenhang gebracht werden, um eine numerische nicht zu aufwändige Bearbeitung zu ermöglichen.

Zwei sich zum Zeitpunkt t nicht schneidende Kugeln, deren Orte und Geschwindigkeiten eine spätere Kollision gewährleisten, werden zu genau zwei Zeiten $t_{ij}^{1,2}$ den Abstand σ zueinander einnehmen: Einmal, wenn sie sich zuerst begegnen und, gesetzt den Fall, dass sie sich ohne Stoß durchdringen können, ein weiteres Mal, wenn sie einander wieder

[§]Die Notwendigkeit, die Simulationsbox in Unterzellen zu unterteilen, wird zu einem späteren Zeitpunkt erläutert. An dieser Stelle sei lediglich bemerkt, dass diese Technik verwendet wird, um Rechenzeit einzusparen.

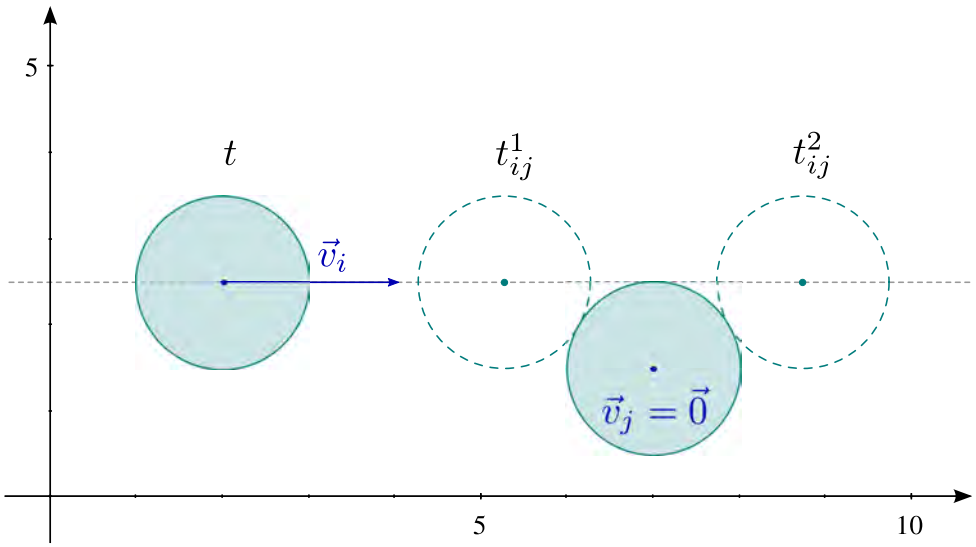


Abbildung 1.4: Die Darstellung veranschaulicht die beiden möglichen Lösungen für t_{ij} aus Gleichung (1.63). Es handelt sich dabei um die beiden Zeitpunkte, zu denen der Abstand zwischen den beiden Teilchen gleich ihrem Durchmesser ist, sie sich also berühren. Wird davon ausgegangen, dass sich die Teilchen durchdringen können, sind eben die beiden dargestellten Konstellationen möglich. Zum einen wenn sich die Teilchen nach einer vorherigen Annäherung erstmalig treffen und zum anderen, wenn sie sich nach einer möglichen Durchdringung wieder verlassen. Innerhalb unserer Simulationen ist jedoch stets eine Durchdringung ausgeschlossen, weshalb als Stoßzeit jeweils die geringere zu wählen ist.

verlassen; siehe dazu auch Abb. 1.4. Der zweite Fall ist für uns nicht von Interesse, da wir den Stoß natürlich vollziehen wollen, und eine Durchdringung somit ausgeschlossen ist. Mit dieser Betrachtung lässt sich argumentieren, dass stets die kleinere der Lösungen aus Gleichung (1.63) die für uns relevante darstellt. Es bleibt noch zu bemerken, dass auch für b_{ij} eine Einschränkung existiert. Aus Abb. 1.5 lässt sich ersehen, dass die Projektion des Relativgeschwindigkeitsvektors $\vec{v}_{ij}$ stets antiparallel zum Verbindungsvektor $\vec{r}_{ij}$ zweier Teilchen steht, wenn diese sich einander annähern und parallel, wenn sie sich voneinander entfernen. Da $b_{ij} = \vec{r}_{ij} \cdot \vec{v}_{ij}$ das Skalarprodukt dieser Vektoren ist, wird der Wert im Falle einer Annäherung und somit einer möglichen Kollision stets negativ sein. Ist b_{ij} positiv, entfernen sich die Teilchen voneinander, und die Berechnung des Stoßzeitpunktes ist hinfällig. Ein Stoß ist aber auch bei negativem b_{ij} nicht garantiert. Es bedeutet lediglich, dass sich die beiden Teilchen in diesem Moment, bedingt durch ihre Positionen und Geschwindigkeiten, einander annähern. Es ist aber immer noch möglich, dass sie einander verfehlen, oder vor dem Stoß durch ein drittes Teilchen bzw. ein anderes Ereignis abgelenkt werden; siehe Abb. 1.6. Mathematisch tritt erster Sachverhalt durch einen negativen Radikant in Gleichung (1.63) in Erscheinung. In diesem Fall existiert somit keine reelle Lösung für die quadratische Gleichung, ein Stoß findet also nicht statt.

Die Idee ist nun alle potentiellen, in der Zukunft liegenden, Stöße zu ermitteln und das System bis zu dem Zeitpunkt des zeitlich nächsten Stoßes zu propagieren. Der Stoß kann

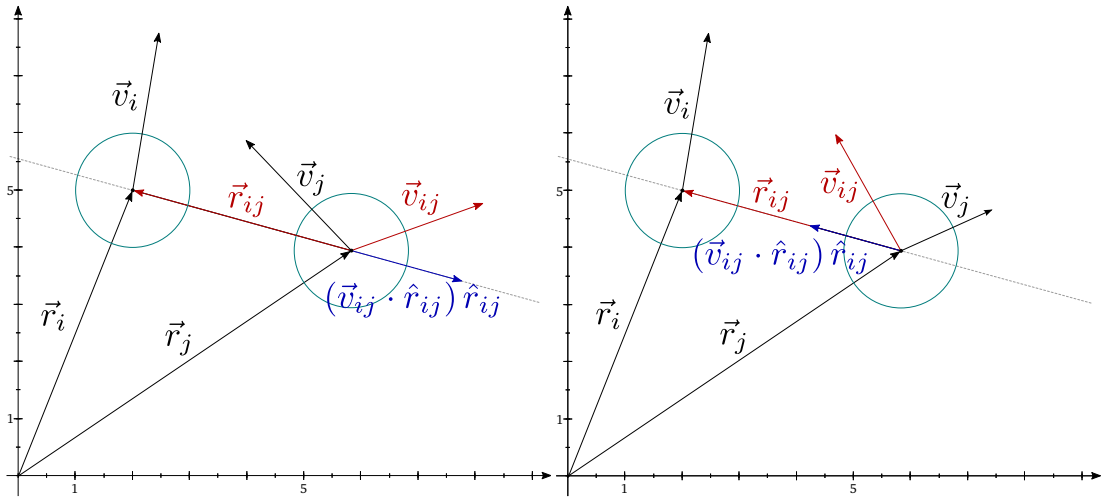


Abbildung 1.5: Diese Abbildungen zeigen in vektorieller Darstellung zwei Kugeln an den Orten $\vec{r}_i$ und $\vec{r}_j$ mit den Geschwindigkeiten $\vec{v}_i$ und $\vec{v}_j$. Die Vektoren $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ und $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$ stellen den Verbindungsvektor und die Relativgeschwindigkeit der beiden Teilchen dar. Des Weiteren ist $\vec{v}_{ij} \cdot \hat{r}_{ij}\,(= b_{ij}/\|\vec{r}_{ij}\|)$ mit $\hat{r}_{ij} = \vec{r}_{ij}/\|\vec{r}_{ij}\|$ die Projektion der Relativgeschwindigkeit auf den Verbindungsvektor. Es lässt sich der Abbildung entnehmen, dass die vektorielle Projektion $(\vec{v}_{ij} \cdot \hat{r}_{ij})\,\hat{r}_{ij}$ stets antiparallel zum Verbindungsvektor $\vec{r}_{ij}$ zweier Teilchen steht, wenn diese sich einander annähern und parallel, wenn sie sich voneinander entfernen.

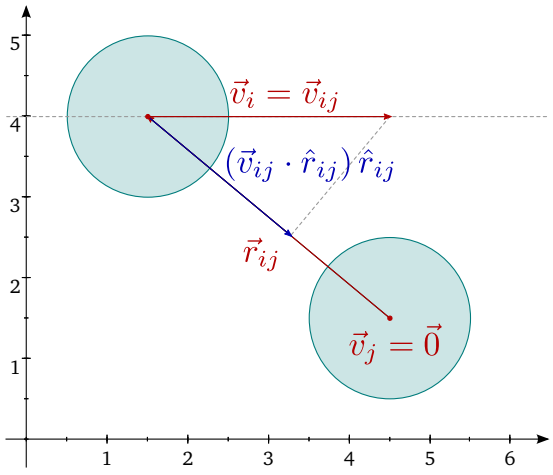


Abbildung 1.6: Die Abbildung zeigt die Möglichkeit einer Antiparallelität von $\vec{r}_{ij}$ und $(\vec{v}_{ij} \cdot \hat{r}_{ij})\,\hat{r}_{ij}$, ohne dass es zwischen den Teilchen zu einer Kollision kommen kann. Dies entspricht mathematisch dem Fall, dass zwar einerseits $b_{ij} < 0$, aber andererseits der Radikant aus Gleichung (1.63) ebenfalls kleiner Null ist, und somit keine reelle Lösung existiert.

dann vollzogen und die Berechnung von vorne begonnen werden. Im Prinzip würde diese Herangehensweise funktionieren, aber die Berechnung aller möglicher Stoßpaarungen ist ein Problem der Größenordnung $\mathcal{O}(N^2)$. Im Falle periodischer Randbedingungen und geringer Dichte kommt noch hinzu, dass alle umliegenden Spiegelteilchen mitbetrachtet werden müssen, um sicher den zeitlich näherliegenden Stoß zu erhalten. Damit erhöht sich der Rechenaufwand noch einmal um den Faktor 9 im Fall eines zweidimensionalen Systems und um Faktor 27 bei einem dreidimensionalen. Das Lösen der quadratischen Gleichung ist jedoch rechenaufwändig genug, um diesen Ansatz sofort zu verwerfen.

Um Rechenzeit zu sparen, wird die Simulationsbox in Zellen mit einer Kantenlänge von mindestens einem Teilchendurchmesser eingeteilt.[14] Damit ist sichergestellt, dass ausschließlich Teilchen benachbarter Zellen miteinander stoßen können. Es muss lediglich ein weiteres Ereignis eingebracht werden, nämlich der Übertritt eines Teilchens von einer Zelle zur nächsten. In diesem Schema können periodische Randbedingungen genauso wie harte Wände in einfacher Weise behandelt werden, und die Betrachtung möglicher Spiegelteilchen entfällt ebenso wie die Notwendigkeit der Rückfaltung eines die Simulationsbox verlassenden Teilchens. Die Berechnung der Paarungen beschränkt sich somit einzig auf die Nachbarzellen. Ist die Zellgröße beispielsweise so gewählt, dass sich im Mittel lediglich ein einziges Teilchen in jeder Zelle befindet, müssen pro Teilchen im Mittel lediglich 26 bzw. 8 mögliche Stöße berechnet werden und nicht $N - 1$. Der Zeitpunkt des Zellübertritts muss natürlich für jedes Teilchen berechnet und der Übertritt vollzogen werden. Der dadurch entstehende Mehraufwand wird aber durch die Ersparnis bei der Berechnung der Stoßpaarungen mehr als kompensiert.

Weiterhin trägt zur Rechenzeitersparnis bei, dass nach einem Ereignis nicht alle zuvor berechneten Ereignisse invalidiert werden. Nach einem Stoß beispielsweise müssen lediglich die zukünftigen Stöße und Zellübertritte derjenigen Teilchen neu berechnet werden, an denen Teilchen des gerade vollzogenen Stoßes beteiligt waren. Alle anderen Ereignisse bleiben vorerst gültig. Die Organisation der Ereignisse findet im sogenannten Ereigniskalender statt, der üblicherweise die Form eines binären Baumes annimmt. Die Vorzüge und Nachteile der verschiedenen Varianten binärer Bäume und anderer Datenstrukturen wurde in der Literatur ausführlich diskutiert.[15,16] Als Beispiel für eine Implementierung eines binären Baumes sei auf die in der C++ *Standard Template Library* (STL) definierten Klassen *Map* und *Multimap* verwiesen.[17]

Weitere Informationen zu diesem Thema und auch zu verwendeten Algorithmen können in Artikeln von Lubachevsky,[18] Marin, Risso und Cordero[19] sowie Isobe[20] gefunden werden. Eine parallele Implementierung wurde von Miller und Luding beschrieben.[21]

Kapitel 2

Untersuchung großer Fluktuationen im Lorentz-Gas

2.1 Motivation

Große Fluktuationen um stationäre Zustände sind selten, aber verantwortlich für eine Reihe von Phänomenen wie beispielsweise Phasenübergänge, DNA-Mutationen und chemische Reaktionen. Sie sind zumindest mitverantwortlich für das nicht umkehrbare Verhalten makroskopischer Systeme mit umkehrbarer mikroskopischer Dynamik. Eine grundlegende Fragestellung betrifft die Beziehung zwischen Anregungs- und Abklingpfaden großer Fluktuationen. Experimentell wurden Asymmetrien zwischen diesen Pfaden bei Störungen von analogen elektronischen Geräten beobachtet.[22] Die Fluktuationstheorie für stationäre Zustände von Nichtgleichgewichtssystemen von Onsager, Machlup, Bertini, De Sole, Gabrielli, Jona-Lasinio und Landim[23–25] sagt eine Asymmetrie in den Anregungs- und Abklingpfaden allgemein bei großen Fluktuationen in mesoskopischen Systemen voraus. Diese Theorie von A. Gamba und L. Rondoni wurde anhand einer Computersimulation an einem modellhaften Elektronengas untersucht.[26] Es handelt sich hierbei um das sogenannte Lorentzgas, ein nicht relativistisches und nicht quantenmechanisches Modell der freien Elektronen in einem Metall. Die damalige Untersuchung konnte keine Asymmetrien in den Fluktuationen nachweisen, jedoch verblieben Zweifel aufgrund des schlechten Signal-zu-Rausch-Verhältnisses. Andererseits ließe sich die Abwesenheit von Asymmetrien im Lorentzgas damit erklären, dass die einzelnen Teilchen, also die Elektronen, nicht untereinander wechselwirken, was eine Forderung der Theorie ist.

Die vorliegende Arbeit soll die restlichen Zweifel ausräumen. Zu diesem Zweck wurde die gesamte Stoßdynamik sowie auch sämtlicher Programmcode zur Analyse von Grund auf neu entwickelt, bei der Analyse wurde nicht nur die globale Stromstärke betrachtet; Abb. 2.4 zeigt die Bereiche, in welche die Simulationszelle zwecks Analyse unterteilt wurde. Ebenso kamen mehrere Definitionen zur Fluktuationsdetektion zum Einsatz.

Wie auch in der vergangenen Untersuchung wurden in unserer Simulation keine Anzeichen auf eine Asymmetrie in der Form der Fluktuationen entdeckt, allerdings ist die Statistik unserer Auswertung deutlich verbessert.

2.2 Aufbau der Simulation

Es soll ein freies, klassisches Elektronengas in einem zweidimensionalen Ionengitter simuliert werden. Die Einheitszelle hat die Form eines Rhombus mit den Winkeln 60° und 120°, siehe Abb. 2.1, 2.2 und 2.3. An den Eckpunkten sitzt jeweils ein Atomrumpf als harter Streuer. Die Zelle wird zur Messung und Auswertung in fünf Bereichen separat betrachtet, siehe Abb. 2.4. In jedem dieser Bereiche wird zu jedem Zeitschritt die aktuelle Stromstärke bestimmt und als Zeitreihe aufgezeichnet.

Parameter zur Steuerung der Simulation

Die Simulation wird über einen relevanten Satz von sechs Parametern gesteuert, hinzu kommen noch vier weitere, die mit der eigentlichen Simulation nicht direkt in Verbindung stehen. Im Einzelnen sind dies:

spacing (s) Beschreibt den Rand-Rand Abstand benachbarter Streuer. Mögliche Werte liegen im Bereich $s \in \left(0, 2(\frac{2}{\sqrt{3}} - 1)\right) \approx (0, 0.309]$; siehe dazu auch Abb. 2.3 und Abb. 2.5. Die Streuer haben einen Durchmesser von 1; anders ausgedrückt werden reduzierte Einheiten[1] benutzt, deren Längeneinheit der Durchmesser der Streuer ist.

field (ϵ) Ist ein Maß für die Stärke des verwendeten äußeren elektrischen Felds $\vec{E} = (\epsilon, 0)$. Sinnvolle Werte in reduzierten Einheiten liegen hier im Bereich $\epsilon \in [0, 2.5]$.

deltat (Δt) Gibt die Größe des Zeitschritts in reduzierten Einheiten wieder. Der Wert $\Delta t = 0.001$ hat sich als geeignet zur Energieerhaltung erwiesen.

nstep (number of steps) Steht für die Anzahl an Zeitschritten, über welche jede ermittelte Elektronentrajektorie gemessen werden soll.

ntraj (number of trajectories) Steht für die Anzahl der insgesamt zu berechnenden Trajektorien, bzw. für die Gesamtzahl der Elektronen im System.

width Stellt die halbe Breite derjenigen Streifen dar, welche die Zelle in unterschiedliche Bereiche unterteilt; siehe Abb. 2.4. Der von uns gewählte Wert von $w = 0.25944$ sorgt jeweils für gleiche Flächen der Bereiche innerhalb und außerhalb der Streifen.

seed Initialisiert den Zufallszahlengenerator und damit die zufällige Anfangsrichtung der Geschwindigkeit; danach erfolgt die Entwicklung deterministisch. Eine Änderung

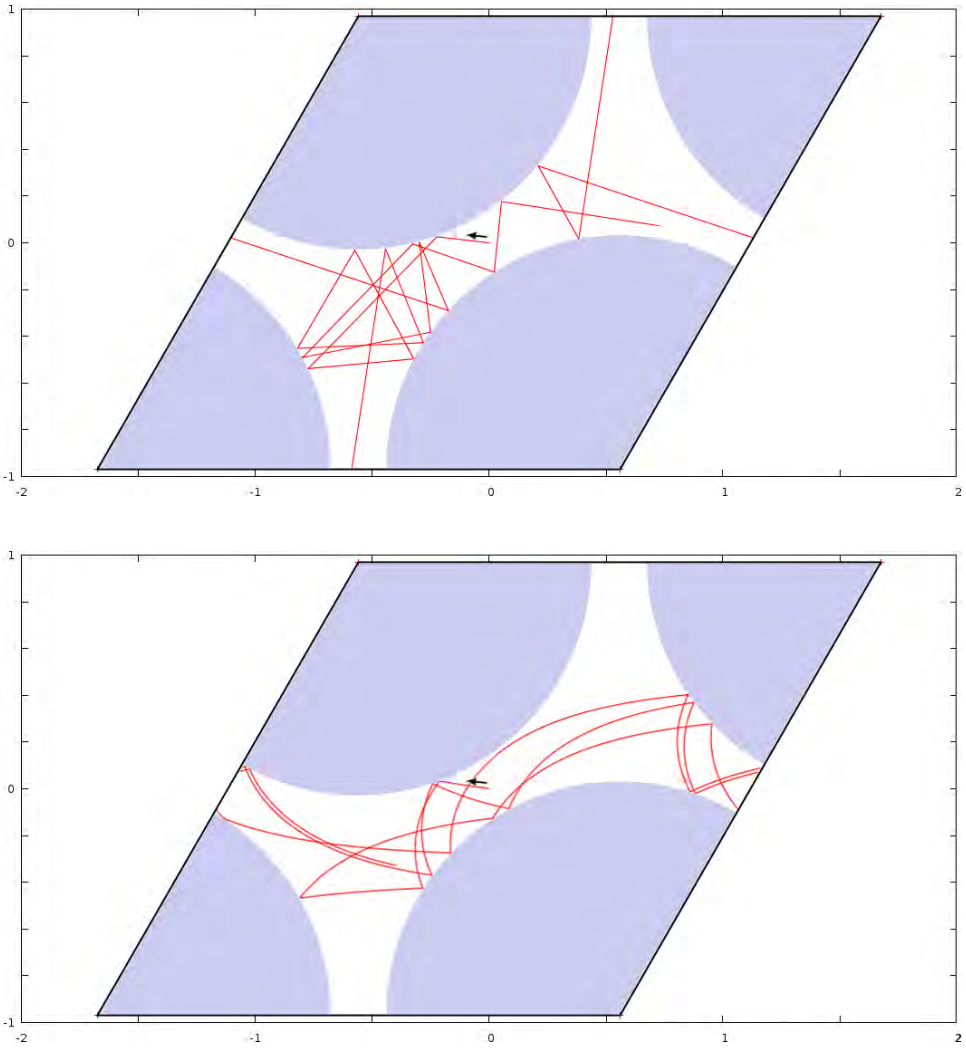


Abbildung 2.1: Diese Abbildung zeigt exemplarisch die berechneten Elektronenbahnen in der Einheitszelle. Dargestellt sind jeweils die ersten 11 000 berechneten Zeitschritte von $\Delta t = 0.001$ für eine Simulation ohne äußeres elektrisches Feld (oben) und bei einem starken Feld (unten) mit $\epsilon = 2.3$. Deutlich zu erkennen sind sowohl die gebogenen Flugbahnen im Feld als auch die periodischen Randbedingungen in beiden Fällen (das Teilchen verlässt die Box durch eine Seite und betritt sie sofort wieder durch die gegenüberliegende).

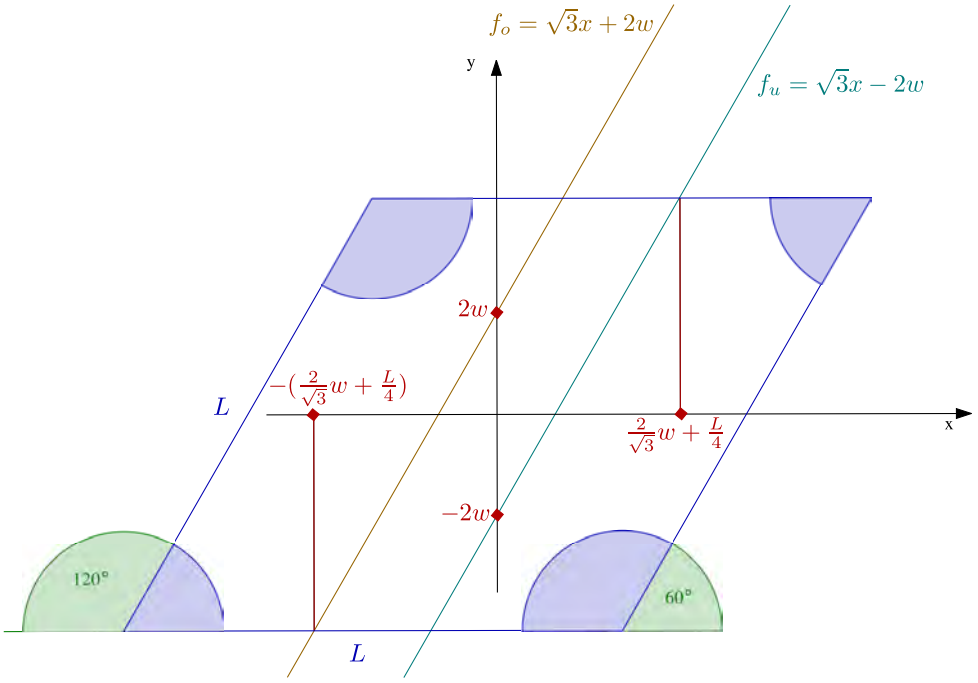


Abbildung 2.2: Eins der Gebiete, in welche die Simulationsbox aufgeteilt wurde, hat die Form eines schrägen Streifens parallel zu zwei Seiten des Parallelogramms und mittig durch die beiden anderen. Um zur Laufzeit entscheiden zu können, ob sich ein Elektron innerhalb dieses Streifens oder außerhalb aufhält, müssen die begrenzenden Geraden bekannt sein. Eine Betrachtung der hier dargestellten Ergebnisse der Berechnungen aus Abb. 2.4 ergibt die Grenzen $x \in \left[-(\frac{2}{\sqrt{3}}w + \frac{L}{4}), (\frac{2}{\sqrt{3}}w + \frac{L}{4})\right]$ und $y \in \left[\sqrt{3}x - 2w, \sqrt{3}x + 2w\right]$.

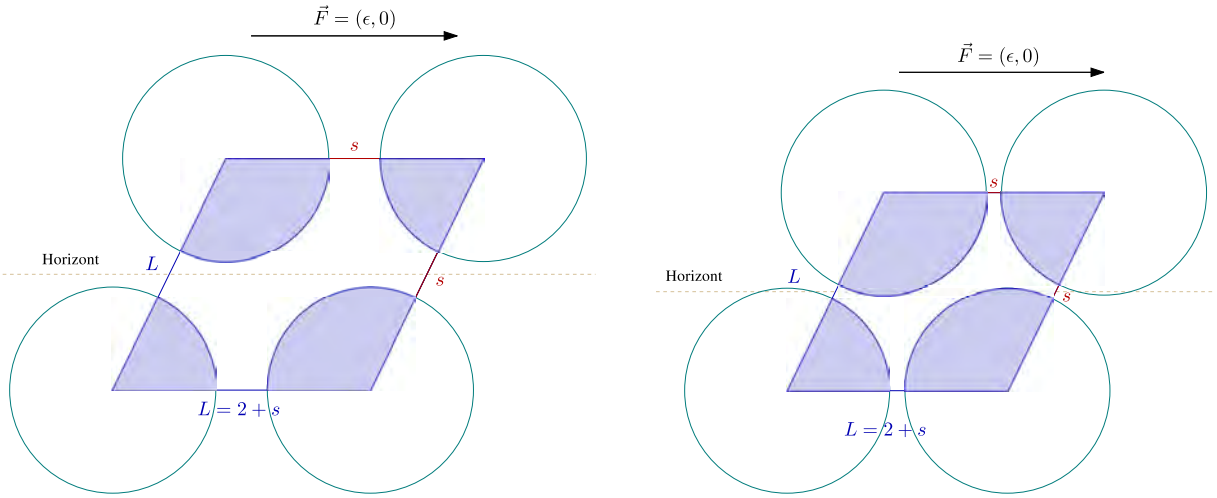


Abbildung 2.3: Die links dargestellte Simulationsbox zeigt einen sogenannten offenen Horizont, d. h. es ist möglich eine Gerade durch die Box zu legen, ohne die Streuer zu schneiden. Während der Simulation wären dann periodische Elektronenbahnen möglich. Aus diesem Grund muss der Parameter s (Randabstand benachbarter Streuer) passend gewählt werden, um einen geschlossenen Horizont (rechte Abbildung) zu gewährleisten. Die obere Grenze hierfür liegt bei $s = 2(\frac{2}{\sqrt{3}} - 1) \approx 0.309$, siehe Abb. 2.5.

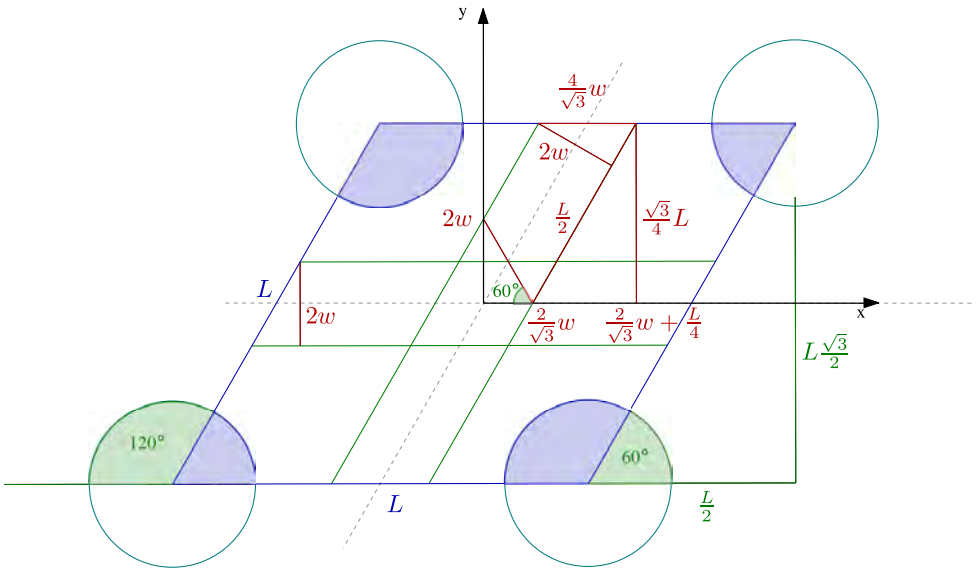


Abbildung 2.4: Diese Abbildung zeigt Bereiche mit der Breite $2w$, welche die Simulationsbox unterteilen (ein horizontaler Streifen und ein schräger Streifen). Die Datenauswertung erfolgt getrennt nach innerhalb bzw. außerhalb des horizontalen bzw. schrägen Streifens und der gesamten Simulationsbox, insgesamt also in fünf Bereiche. Die Begrenzungen des horizontalen Streifens sind direkt der Zeichnung zu entnehmen. Da sich das Elektron nicht außerhalb der Simulationsbox aufhalten kann, genügt $y \in [-w, w]$. Die notwendige Berechnung der Begrenzungen des schrägen Streifens ist ebenfalls dieser Zeichnung zu entnehmen. Das Ergebnis hierzu ist in Abb. 2.2 zusammengefasst.

diesen Wertes erzeugt somit andere Trajektorien, welche allerdings die selben Mittelwerte liefern.

Parameter für die Analyse

vacfsize (velocity autocorrelation function size) Gibt die Länge, gemessen in Anzahl an Zeitschritten, der Geschwindigkeits-Autokorrelationsfunktion (VACF) an. Es ist sinnvoll, tskip oder 2×tskip zu nehmen, weil gemäß der Definition von tskip (s.u.) dann die VACF auf Null gefallen ist.

msdsize (mean square displacement size) Gibt die Länge, ebenfalls in Zeitschritten, der mittleren quadratischen Abweichung (MSD) an. Die Obergrenze ist nstep/tskip.

tskip (time skip) Sowohl für die VACF als auch die MSD werden nicht alle Zeitschritte verwendet, sondern nur solche, die so weit auseinander liegen, dass sie als unkorreliert betrachtet werden können. Die VACF beginnt alle tskip Zeitschritte, während bei der MSD nur jeder tskip'te Zeitschritt gewertet wird. Unter Annahme einer Gauß'schen Statistik setzt man tskip gleich zwei mal der Korrelationszeit, wobei die Korrelationszeit zwei mal das Zeitintegral des Quadrates einer Autokorrelationsfunktion (z.B. der VACF) ist.[1] In Formeln: $t_{\text{skip}} = 2t_{\text{corr}}$, $t_{\text{corr}} = 2\int_0^\infty c^2(t)dt$, mit

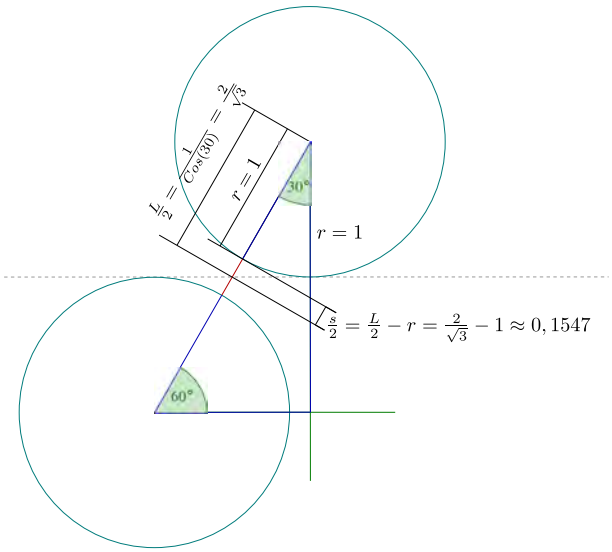


Abbildung 2.5: Illustration zur Berechnung des maximalen Abstandes zweier Streuzentren, so dass gerade der Horizont gerade noch geschlossen bleibt. Die Details der Berechnung können der Zeichnung entnommen werden. Es ergeben sich die möglichen Grenzen für den Parameter s zu $s \in \left(0, 2 \cdot \left(\frac{2}{\sqrt{3}} - 1\right)\right) \approx (0, 0.309]$.

$c(t) = \text{ACF}.$

Die VACF $c\,(t)$ und die MSD $\sigma^2\,(t)$ werden nicht bei jedem Simulationslauf mitberechnet, sondern lediglich für solche ohne angelegtes äußeres elektrisches Feld zur Validierung der Simulation. In späteren Produktionsläufen wurde die Berechnung dieser Funktionen ausgeschaltet, da sie nicht nötig und zeitintensiv ist. Der Diffusionskoeffizient D lässt sich sowohl aus der Green-Kubo-Relation über die Hälfte des Integrals der Geschwindigkeitskovarianzfunktion $C\,(t)$ bestimmen, wie auch aus der Einstein-Relation über dem Viertel der Steigung des MSD. In Formeln $D = 1/d \int_0^\infty C\,(t)\,dt = (1/2d)\ d\sigma^2\,(t)\,/dt$ mit $d = 2$. Im vorliegenden Fall ist der erste Wert der Geschwindigkeitskovarianzfunktion $C\,(0) = 1$, dementsprechend kann die VACF zur Berechnung der Diffusionskoeffizienten benutzt werden, wegen $c\,(t) = C\,(t)\,/C\,(0) = C\,(t)$. Zu sehen sind die berechneten VACF und MSD in Abb. 2.6.

Da im Modell des Lorentzgases die Elektronen nicht wechselwirken, ist es möglich, die einzelnen Trajektorien der Elektronen getrennt voneinander zu berechnen. Daraus ergibt sich eine einfache und effiziente Möglichkeit, die Simulation zu parallelisieren, indem auf einem Parallelrechner von der Gesamtzahl jeweils nur ein Teil der Trajektorien pro CPU-Kern berechnet werden. Der Programmcode ist solcherart strukturiert, dass er sowohl für ein Einprozessorsystem als auch für einen Parallelrechencluster mit einer beliebigen Anzahl von Rechenkernen zu kompilieren ist. Die zur Parallelisierung benutzte Schnittstelle ist das Message Passing Interface (MPI).[27]

Ein definierter Zeitschritt ist zur Aufzeichnung der geforderten Zeitreihe unabdingbar. Aus diesem Grund wurde entschlossen, die gesamte Simulation zeitschrittgesteuert zu im-

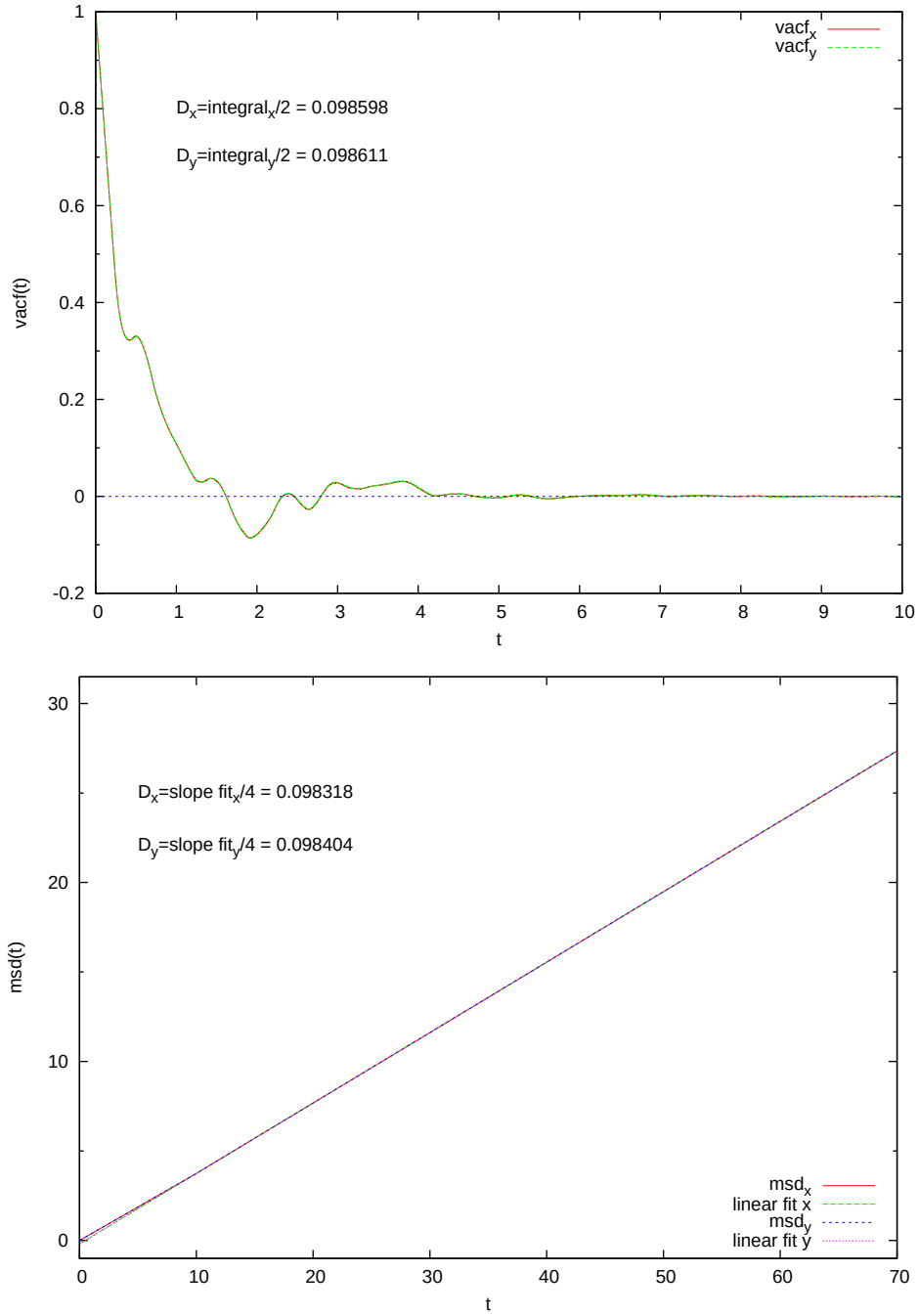


Abbildung 2.6: Diese Abbildung zeigt oben die Geschwindigkeitsautokorrelation (velocity autocorrelation function, VACF) und unten die mittlere, quadratische Abweichung (mean square displacement, MSD) für ein Elektronengas ohne äußeres angelegtes Feld ($\epsilon = 0$). Die Ergebnisse sind konsistent, denn sowohl VACF, als auch MSD sind wie erwartet unabhängig von der Raumrichtung, und der Diffusionskoeffizient aus der Hälfte des Integrals der VACF (Green-Kubo) entspricht einem Viertel der linearen Steigung der MSD (Einstein-Relation).

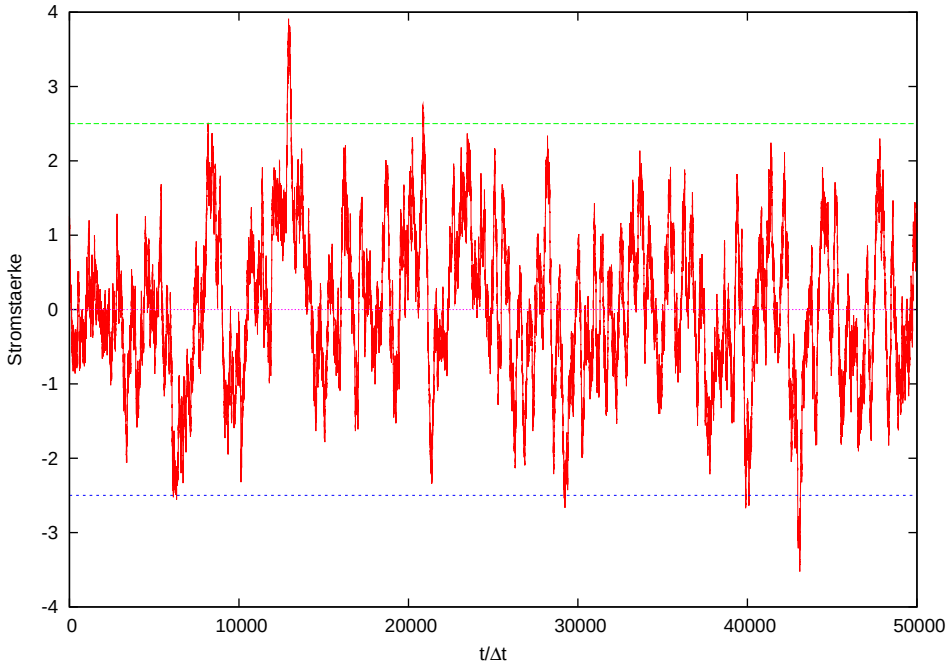


Abbildung 2.7: Die Abbildung zeigt die ersten 50 000 Zeitschritte einer Lorentzgassimulation von 10 000 Elektronen ohne ein wirkendes äußeres elektrisches Feld. Die waagerechten Linien bezeichnen die Nulllinie, sowie $\pm 2.5\sigma$. Eine Auslenkung um mindestens 2.5σ wird als eine Fluktuation im Sinne der zu prüfenden Theorie angesehen.

plementieren. Ein ereignisgesteuerter Ansatz, mit Schritten von Stoß zu Stoß, ist zwar prinzipiell möglich, allerdings hier nicht unbedingt zweckmäßig und vom Aufwand her nicht unbedingt günstiger. Dies ist in der Existenz des äußeren Feldes begründet, welches die ansonsten geraden und leicht handhabbaren Teilchenbahnen zu Parabeln deformiert. Die Berechnung der Stoßzeiten bei Parabelbahnen würde dann die Lösung eines Problems vierter Ordnung bedeuten, was in sich schon recht kompliziert ist. Da die Statistiken zu festen Zeitschritten erhoben werden müssen, ist es ohnehin notwendig das System von Messereignis zu Messereignis zu propagieren. Der Einfachheit wegen kann dann auch direkt die gesamte Simulation zeitschrittgesteuert erfolgen. Der hier gewählte Ansatz benutzt den, in den Abschnitten 1.1.3 und 1.1.4 vorgestellten, Velocity-Verlet-Integrator. Hinzu kommt noch ein isokinetischer Thermostat, um die Energie, welche das äußere Feld in das System pumpt, wieder abzuführen, da sich ansonsten kein Gleichgewicht einstellen kann.[28] Insgesamt sind die Laufzeiten gering genug, als das sich eine Optimierung mit dem damit verbundenen Mehraufwand bei der Programmerstellung auszahlen würde.

Die Ausgabe der Simulation besteht aus einer Zeitreihe von Stromstärken für jeden der fünf definierten Bereiche der Simulationszelle. Einen Eindruck wie diese Reihen aussehen soll Abb. 2.7 vermitteln. Dort sind für den Bereich der gesamten Zelle die ersten 50 000 Zeitschritte eines Simulationslaufs mit 10 000 Elektronen bei ausgeschaltetem externen Feld dargestellt. Die Werte werden normiert auf einen Mittelwert von $\mu = 0$ und eine Standardabweichung von $\sigma = 1$, um die weitere Betrachtung einem Algorithmus leichter

zugänglich zu machen.

Wir interessieren uns für große Auslenkungen vom Mittelwert der Zeitreihe und insbesondere die Symmetrie des ansteigenden und abfallenden Pfades innerhalb einer solchen Fluktuation.

2.3 Aufbau und Test der Analysesoftware

Das Interesse richtet sich auf große Abweichungen vom Gleichgewichts- bzw Mittelwert und insbesondere auf die zeitliche Symmetrie oder Asymmetrie einer solchen Fluktuation. Die Analyse besteht nun darin, zunächst den Mittelwert und die Standardabweichung einer gegebenen Zeitreihe zu normieren, damit die weitere Bearbeitung vereinfacht wird. Hier wird sinnvollerweise für den Mittelwert $\mu = 0$ und die Standardabweichung $\sigma = 1$ gewählt, und der modifizierte Prozess, gesichert. Im Weiteren wird der normierte Prozess nach Fluktuationen abgesucht, die eine gewisse Schwelle überschreiten. Diese wurde in Übereinstimmung mit vergangenen Veröffentlichungen[26,29] auf das zweieinhalbfache der Standardabweichung, in unserem Fall also auf 2.5 gesetzt. Eine Fluktuation bestimmt sich nun aus zwei Durchgängen durch diese Schwelle. Die beiden Durchgänge müssen dabei ausgehend vom mittleren Bereich durch eine Schwelle und wieder zurück stattfinden. Die Mitte der Fluktuation wird von uns auf zweierlei Arten festgelegt, nämlich einmal als Mitte und einmal als Position des Maximums/Minimums zwischen den beiden Schwellendurchgängen. Die Analyse findet getrennt nach positivem/negativen Teil des Prozesses und aufgeteilt für die beiden Mittendefinitionen statt. Daraus ergeben sich vier zu untersuchende Fälle, die einzeln betrachtet werden.

Die gefundenen Fluktuationen werden mit einer Umgebung von einigen Zeitschritten um den definierten Mittelpunkt in ein Zweidimensionales Histogramm eingetragen. Die genaue Größe der Umgebung ist ein Eingabeparameter der Analyse und wird meist auf 100 oder mehr Zeitschritte gesetzt. Da die Fluktuationen in einem Histogramm jeweils die gleiche Mittendefinition aufweisen und auch sonders positiv oder negativ sind, ergibt sich im Histogramm ein Bild, welches einem Bergmassiv mit einer ausgeprägten Spitze in der Mitte und aufweitenden Ausläufern ähnelt; siehe dazu die Abbn. 2.13 und 2.16 oben links. Der Grat dieses Gebirges stellt in etwa die mittlere oder auch wahrscheinlichste Fluktuation dar. Um eine genauere Mittelung zu erreichen, wird noch zusätzlich entlang eines Zeitschritts gemittelt. Der resultierende Grat ergibt bei ausreichender Statistik ein sehr glattes Bild einer mittleren Fluktuation.

Um ein gewisses Vertrauen in die Ergebnisse des Programms zur Auswertung der Simulation zu gewinnen, werden fünf Zeitreihen analysiert, welche von vorne herein bekannt und einfach aufgebaut sind. Alle Reihen bestehen aus einer Abfolge von positiven und negativen Fluktuationen, welche jeweils eine lineare steigende und eine lineare fallende Flanke aufweisen. Die erste Reihe, siehe Abb. 2.8, besteht sonders aus symmetrischen

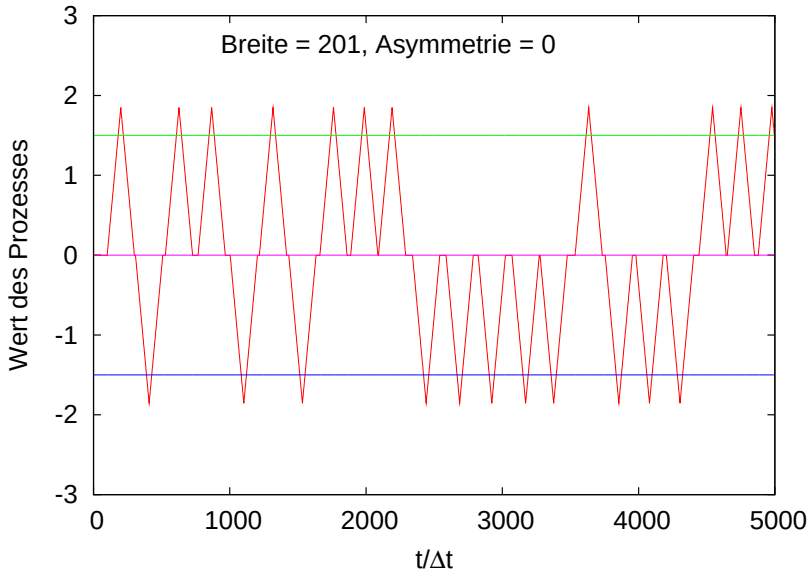


Abbildung 2.8: Die Abbildung zeigt die ersten 5 000 Zeitschritte eines künstlich erzeugten Prozesses. Die Breite jeder Fluktuation beträgt 201 Zeitschritte. Es ist keine Asymmetrie vorhanden.

Fluktuationen mit einer Breite von 201 Zeitschritten und gleicher Amplitude. Bei der zweiten in Abb. 2.9 gezeigten Reihe wurde die Breite zufällig variiert. Dazu wird jeweils ein einhundertseitiger Würfel (W_{100}) gerollt, das Ergebnis mit zwei multipliziert und 201 addiert, kurz: Breite $= 2W_{100} + 201$. Bei den Reihen drei, vier und fünf wurde schließlich eine Asymmetrie in die Fluktuationen eingebracht. Dazu wurde jeweils der Ast rechts des Maximums/Minimums einer Fluktuation um einen, zehn bzw. einhundert Zeitschritte verbreitert. Siehe dazu Abbn. 2.10, 2.11 und 2.12.

An dieser Stelle sollen zum Test nur die Darstellungen für die positiven Fluktuationen gezeigt werden. Weiterhin gilt für die Mittendefinition das Maximum zwischen den Schwellendurchgängen. Die zweidimensionalen Stapelhistogramme werden in Abb. 2.13 dreidimensional dargestellt. Die einem Gebirge ähnelnde Struktur ist jeweils gut zu erkennen. Eine Ausnahme ist hier natürlich das Histogramm für den ersten Testprozess aus identischen Fluktuationen, da dieser eine Scharfe Kante ergeben muss. Die sichtbare Zahnung der Kante ist im Übrigen lediglich ein Effekt, der auf der Art der Abbildung und des Binnings beruht und grundsätzlich ohne weitere Bedeutung. Die wahrscheinlichste Form einer Fluktuation kann, für jeden Prozess, aus dem zugehörigen „Gebirge“ ermittelt werden. Im Rahmen dieser Arbeit geschieht dies auf zweierlei Weise. Zum einen kann der Grat der Verteilung, also der Verlauf des Maximums entlang der Zeitschritte als wahrscheinlichste Form einer Fluktuation angesehen werden. Aus dieser Interpretation ergibt sich aber ein stark verrauschtes Abbild für die wahrscheinlichste Fluktuation. Zum anderen kann aber auch zu jedem Zeitschritt der Breite des Gebirges der Mittelwert der Einträge gebildet werden. In anderen Worten ist das jeweils der Mittelwert der Einträge in einem

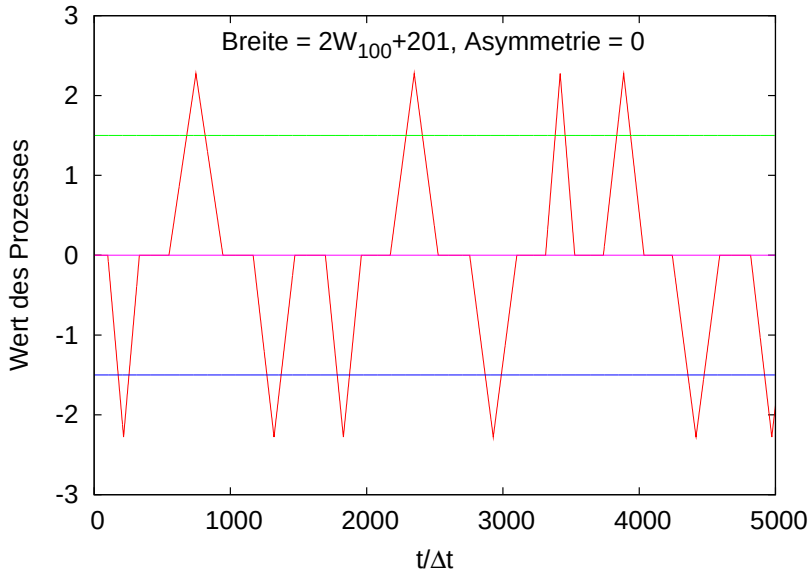


Abbildung 2.9: Die Abbildung zeigt die ersten 5 000 Zeitschritte eines künstlich erzeugten Prozesses. Die Breite jeder Fluktuation beträgt $2W_{100} + 201$ Zeitschritte. Es ist keine Asymmetrie vorhanden.

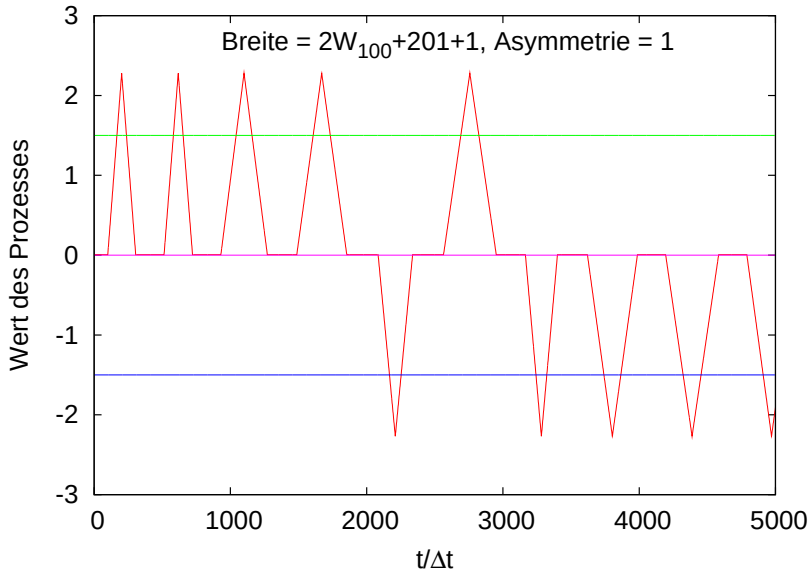


Abbildung 2.10: Die Abbildung zeigt die ersten 5 000 Zeitschritte eines künstlich erzeugten Prozesses. Die Breite jeder Fluktuation beträgt $2W_{100} + 201 + 1$ Zeitschritte. Es ist eine Asymmetrie eingearbeitet, indem der rechte Teil einer Fluktuation jeweils um einen Zeitschritt verlängert wurde.

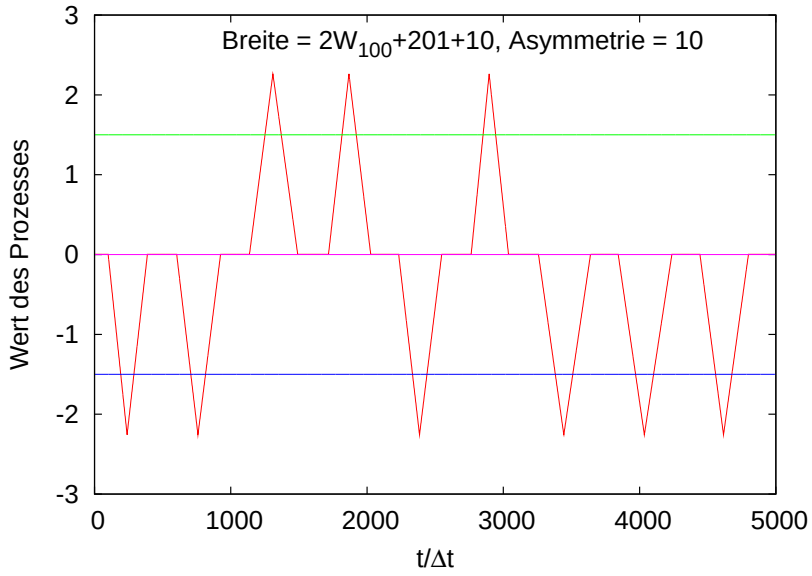


Abbildung 2.11: Die Abbildung zeigt die ersten 5 000 Zeitschritte eines künstlich erzeugten Prozesses. Die Breite jeder Fluktuation beträgt $2W_{100} + 201 + 10$ Zeitschritte. Es ist eine Asymmetrie eingearbeitet, indem der rechte Teil einer Fluktuation jeweils um zehn Zeitschritte verlängert wurde.

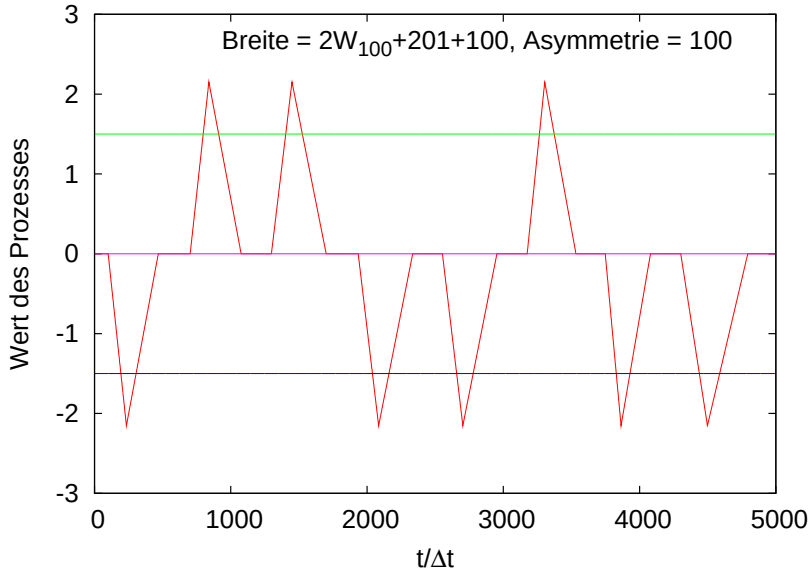


Abbildung 2.12: Die Abbildung zeigt die ersten 5 000 Zeitschritte eines künstlich erzeugten Prozesses. Die Breite jeder Fluktuation beträgt $2W_{100} + 201 + 100$ Zeitschritte. Es ist eine Asymmetrie eingearbeitet, indem der rechte Teil einer Fluktuation jeweils um einhundert Zeitschritte verlängert wurde.

Schnittprofil senkrecht zur Zeitachse des Histogramms. Diese Methode ergibt dann nicht mehr die wahrscheinlichste, sondern die mittlere Form einer Fluktuation. Auf diese Weise kann das statistische Rauschen nahezu vollständig unterdrückt werden. Die Methode die mittlere, anstatt der wahrscheinlichsten Fluktuation zu bestimmen wurde erstmals im Rahmen dieser Arbeit angewendet, während in den vorangegangenen Veröffentlichungen jeweils das Maximum beobachtet wurde.[26,29] Die wahrscheinlichsten und mittleren Fluktuationen sind in Abb. 2.14 dargestellt. Aufgrund der Statistik bei der Festlegung der Breite der Fluktuationen (würfeln) während des Erstellungsprozesses weichen in diesen Beispielen die Breiten der wahrscheinlichsten von denen der mittleren Fluktuationen mitunter stark ab. Dieser Effekt ist aber zu erwarten und kein Artefakt der Analyse an sich.

Um ein Maß für die Asymmetrie der Fluktuationen zu erhalten, kann der Unterschied des Steigungsverhaltens für die Paare der steigenden und fallenden Flanken bestimmt werden. Dazu wird jeweils schrittweise die Steigung zu allen Stellen symmetrisch um die Mitte einer Fluktuation bestimmt, paarweise die Differenz gebildet und über alle Fluktuationen gemittelt. Es sei noch bemerkt, dass mit Steigung jeweils der Betrag der Steigung gemeint ist. Es ergibt sich die Funktion

$$\alpha(i) = \frac{1}{N}\sum_{j=1}^{N}\alpha_j(i) = \frac{1}{N}\sum_{j=1}^{N}\frac{x_j(-i) - x_j(+i)}{i}; \tag{2.1}$$

dabei bezeichnet N die Anzahl der gefundenen Fluktuationen, $x_j(i)$ den Wert der $j-$ten Fluktuation im Abstand i zur Fluktuationsmitte und $\alpha_j(i)$ die Differenz der Steigungen im Abstand i in der $j-$ten Fluktuation. Der Definitionsbereich von $\alpha(i)$ ist $i \in \left[1, \frac{B}{2}\right] \wedge i \in \mathbb{N}$, mit $B =$ Fluktuationsbreite. Die Graphen der Funktion wie sie sich für die Testsysteme ergeben sind in Abb. 2.15 dargestellt. Erwartungsgemäß sind diese für die beiden ersten Systeme konstant null, da diese ohne Asymmetrie konstruiert wurden. Für die verbleibenden Systeme ergibt sich ebenfalls der Erwartung entsprechend jeweils eine waagerechte Gerade, deren Wert dem Unterschied der Flankensteigungen der mittleren Fluktuationen entspricht. Diese Behauptung kann am Beispiel des letzten Systems mit der größten Asymmetrie geprüft werden. Dazu schätzt man die Steigungen der Flanken in Abb. 2.14 unten rechts. Es bezeichnen m allgemein die Steigung und M den gemeinsamen Wert in der Mitte. Seien die Werte an den Rändern mit $x(-100) \approx 0,1$ und $x(+100) \approx 1,1$ abgeschätzt, dann ergibt sich die Steigungsdifferenz zu $\triangle m \approx \left\|\frac{-(M-1,1)}{100}\right\| - \left\|\frac{M-0,1}{100}\right\| = \frac{(M-1,1)-(M-0,1)}{100} = -\frac{1}{100}$. Dieses Ergebnis ist in guter Übereinstimmung mit dem entsprechenden Graphen in Abb. 2.15 unten rechts.

Die analytische Herleitung von $\alpha(i)$ gestaltet sich wie folgt. Wird die zuvor definierte Benennung beibehalten, ergibt sich die Steigung m vom Wert des Mittelpunktes zum Wert an der Stelle i zu $m_i = \frac{M-x(i)}{i}$ und für $-i$ entsprechend zu $m_{-i} = \frac{M-x(-i)}{i}$. Die Differenz

der Steigungen lautet dann $\Delta m = m_i - m_{-i} = \frac{M-x(i)}{i} - \frac{M-x(-i)}{i} = \frac{x(-i)-x(i)}{i} = \alpha\,(i)$. Die Mittelung über alle N Fluktuationen ergibt dann die Funktion $\alpha\,(i)$.

Ein weiteres Mittel, um die Asymmetrie der Fluktuationen zu charakterisieren, ist es ein Histogramm aller Steigungsdifferenzen $\alpha\,(i) = \frac{x(-i)-x(i)}{i}$ aller Fluktuationen zu bilden. Bei geeigneter Statistik kann dieses Histogramm mit einer Gauß-Kurve gefittet werden, um die Lage des Mittelwerts und die Standardabweichung zu bestimmen. Diese Histogramme sollen an dieser Stelle für die Testsysteme aus Gründen der Übersichtlichkeit nicht gezeigt werden, da diese ohnehin naturgemäß jeweils lediglich einen sehr scharfen Peak enthalten, welcher dem Steigungsunterschied entspricht. Die Streuung der $\alpha-$Werte ist in diesen Fällen nicht geeignet, um sinnvoll einen Gauß-Fit durchzuführen. Beispielhaft ist ein solcher Fit in der folgenden Auswertung in Abb. 2.16 unten links dargestellt.

2.4 Auswertung

Die im Rahmen dieser Schrift berechneten und ausgewerteten Prozesse ergeben sich aus den einhundert möglichen Kombinationen von Feldstärke ($\epsilon = 0.0, 0.5, 1.0, 1.5, 2.0$), betrachtetem Bereich innerhalb der Simulationszelle (siehe Abb. 2.4), sowie Vorzeichen und Mittendefinition der Fluktuationen. Die vier für die Auswertung relevanten Darstellungen zeigt Abb. 2.16. Bei dem dort betrachteten Prozess handelt es sich um die Stromstärke bei ausgeschaltetem äußeren Feld. Der betrachtete Bereich ist die gesamte Simulationsbox. Dargestellt sind alle negativen Fluktuationen, wobei die Mitten als das Minimum zwischen den Schwellendurchgängen definiert wurde. Von besonderem Interesse sind der mittlere Wert für α, der sich mit dem Mittelwert μ des unten links dargestellten Histogramms deckt, sowie der Verlauf der $\alpha-$Werte entlang der Breite der mittleren Fluktuation (unten rechts in Abb. 2.16).

Leider ist die Anzahl der Abbildungen für alle einhundert betrachteten Kombinationen der Parameter zu groß, um diese hier abzudrucken. Dennoch sollen alle Werte für die mittleren α's zumindest in tabellarischer Form angegeben werden. Tab. 2.1 listet alle Kombinationen mit zugehörigen Parametern der Simulation und Zahlenwerten aus den Gauß-Fits auf.

Feld	Bereich	Vorzeichen	Mittendefinition	μ	$\Delta\mu$	σ	$\Delta\sigma$
0.0	gl	negativ	Extremum	0.061	0.013	0.440	0.011
0.0	gl	negativ	Mitte	0.045	0.012	0.429	0.010
0.0	gl	positiv	Extremum	0.009	0.016	0.507	0.013
0.0	gl	positiv	Mitte	0.003	0.016	0.490	0.013
0.0	hi	negativ	Extremum	0.005	0.015	0.419	0.012
0.0	hi	negativ	Mitte	0.001	0.015	0.403	0.012
0.0	hi	positiv	Extremum	-0.045	0.011	0.366	0.009

Feld	Bereich	Vorzeichen	Mittendefinition	μ	$\Delta\mu$	σ	$\Delta\sigma$
0.0	hi	positiv	Mitte	-0.038	0.010	0.360	0.008
0.0	ho	negativ	Extremum	-0.021	0.024	0.668	0.020
0.0	ho	negativ	Mitte	-0.034	0.021	0.653	0.017
0.0	ho	positiv	Extremum	-0.079	0.023	0.667	0.019
0.0	ho	positiv	Mitte	-0.067	0.023	0.653	0.018
0.0	si	negativ	Extremum	0.042	0.027	0.667	0.022
0.0	si	negativ	Mitte	0.033	0.025	0.665	0.020
0.0	si	positiv	Extremum	-0.003	0.025	0.666	0.021
0.0	si	positiv	Mitte	-0.004	0.021	0.649	0.017
0.0	so	negativ	Extremum	0.019	0.016	0.508	0.013
0.0	so	negativ	Mitte	0.017	0.015	0.501	0.012
0.0	so	positiv	Extremum	-0.023	0.020	0.497	0.016
0.0	so	positiv	Mitte	-0.022	0.017	0.478	0.014
0.5	gl	negativ	Extremum	0.044	0.020	0.556	0.016
0.5	gl	negativ	Mitte	0.043	0.024	0.543	0.019
0.5	gl	positiv	Extremum	0.008	0.023	0.571	0.019
0.5	gl	positiv	Mitte	0.000	0.021	0.561	0.017
0.5	hi	negativ	Extremum	0.005	0.009	0.353	0.008
0.5	hi	negativ	Mitte	0.010	0.011	0.345	0.009
0.5	hi	positiv	Extremum	-0.038	0.010	0.384	0.008
0.5	hi	positiv	Mitte	-0.039	0.010	0.373	0.008
0.5	ho	negativ	Extremum	-0.039	0.023	0.711	0.019
0.5	ho	negativ	Mitte	-0.040	0.025	0.685	0.021
0.5	ho	positiv	Extremum	-0.048	0.024	0.747	0.020
0.5	ho	positiv	Mitte	-0.031	0.023	0.721	0.019
0.5	si	negativ	Extremum	0.024	0.022	0.698	0.018
0.5	si	negativ	Mitte	0.022	0.023	0.685	0.019
0.5	si	positiv	Extremum	-0.063	0.027	0.713	0.022
0.5	si	positiv	Mitte	-0.065	0.023	0.699	0.019
0.5	so	negativ	Extremum	0.016	0.021	0.528	0.017
0.5	so	negativ	Mitte	0.013	0.020	0.519	0.016
0.5	so	positiv	Extremum	0.044	0.020	0.526	0.017
0.5	so	positiv	Mitte	0.048	0.019	0.511	0.015
1.0	gl	negativ	Extremum	0.031	0.019	0.503	0.016
1.0	gl	negativ	Mitte	0.009	0.017	0.492	0.014
1.0	gl	positiv	Extremum	-0.095	0.014	0.438	0.011
1.0	gl	positiv	Mitte	-0.094	0.016	0.429	0.013

Feld	Bereich	Vorzeichen	Mittendefinition	μ	$\Delta\mu$	σ	$\Delta\sigma$
1.0	hi	negativ	Extremum	0.020	0.018	0.437	0.015
1.0	hi	negativ	Mitte	0.010	0.018	0.421	0.015
1.0	hi	positiv	Extremum	0.052	0.018	0.410	0.015
1.0	hi	positiv	Mitte	0.025	0.017	0.390	0.014
1.0	ho	negativ	Extremum	0.003	0.018	0.628	0.014
1.0	ho	negativ	Mitte	-0.020	0.019	0.609	0.015
1.0	ho	positiv	Extremum	-0.020	0.018	0.673	0.015
1.0	ho	positiv	Mitte	-0.013	0.019	0.657	0.016
1.0	si	negativ	Extremum	0.025	0.023	0.673	0.019
1.0	si	negativ	Mitte	0.021	0.020	0.662	0.017
1.0	si	positiv	Extremum	0.043	0.023	0.653	0.019
1.0	si	positiv	Mitte	0.037	0.019	0.643	0.016
1.0	so	negativ	Extremum	0.096	0.023	0.584	0.019
1.0	so	negativ	Mitte	0.099	0.022	0.567	0.018
1.0	so	positiv	Extremum	0.054	0.018	0.505	0.015
1.0	so	positiv	Mitte	0.058	0.016	0.504	0.013
1.5	gl	negativ	Extremum	-0.004	0.021	0.595	0.017
1.5	gl	negativ	Mitte	0.006	0.026	0.584	0.021
1.5	gl	positiv	Extremum	-0.030	0.019	0.485	0.015
1.5	gl	positiv	Mitte	-0.032	0.016	0.481	0.013
1.5	hi	negativ	Extremum	-0.054	0.014	0.405	0.012
1.5	hi	negativ	Mitte	-0.056	0.015	0.406	0.012
1.5	hi	positiv	Extremum	0.084	0.019	0.462	0.016
1.5	hi	positiv	Mitte	0.082	0.018	0.441	0.015
1.5	ho	negativ	Extremum	-0.045	0.020	0.647	0.016
1.5	ho	negativ	Mitte	-0.041	0.015	0.628	0.012
1.5	ho	positiv	Extremum	-0.002	0.029	0.760	0.024
1.5	ho	positiv	Mitte	0.001	0.024	0.743	0.019
1.5	si	negativ	Extremum	0.088	0.025	0.725	0.020
1.5	si	negativ	Mitte	0.081	0.032	0.707	0.026
1.5	si	positiv	Extremum	0.083	0.024	0.691	0.019
1.5	si	positiv	Mitte	0.076	0.023	0.679	0.019
1.5	so	negativ	Extremum	0.019	0.016	0.539	0.013
1.5	so	negativ	Mitte	0.022	0.014	0.534	0.011
1.5	so	positiv	Extremum	-0.022	0.021	0.621	0.017
1.5	so	positiv	Mitte	-0.019	0.026	0.598	0.021
2.0	gl	negativ	Extremum	0.051	0.017	0.568	0.014

Feld	Bereich	Vorzeichen	Mittendefinition	μ	$\Delta\mu$	σ	$\Delta\sigma$
2.0	gl	negativ	Mitte	0.044	0.018	0.557	0.014
2.0	gl	positiv	Extremum	-0.073	0.022	0.563	0.018
2.0	gl	positiv	Mitte	-0.056	0.023	0.555	0.019
2.0	hi	negativ	Extremum	0.010	0.023	0.580	0.019
2.0	hi	negativ	Mitte	-0.004	0.018	0.566	0.015
2.0	hi	positiv	Extremum	0.051	0.018	0.538	0.015
2.0	hi	positiv	Mitte	0.060	0.019	0.523	0.015
2.0	ho	negativ	Extremum	0.054	0.026	0.713	0.021
2.0	ho	negativ	Mitte	0.063	0.023	0.698	0.018
2.0	ho	positiv	Extremum	0.028	0.024	0.746	0.020
2.0	ho	positiv	Mitte	0.027	0.024	0.741	0.019
2.0	si	negativ	Extremum	-0.006	0.020	0.729	0.017
2.0	si	negativ	Mitte	-0.009	0.022	0.711	0.018
2.0	si	positiv	Extremum	0.056	0.028	0.738	0.023
2.0	si	positiv	Mitte	0.053	0.027	0.731	0.022
2.0	so	negativ	Extremum	0.020	0.019	0.538	0.015
2.0	so	negativ	Mitte	0.006	0.015	0.523	0.012
2.0	so	positiv	Extremum	-0.054	0.019	0.603	0.015
2.0	so	positiv	Mitte	-0.053	0.021	0.585	0.017

Tabelle 2.1: ollständige Liste aller Gauß-Fits. μ bezeichnet den Mittelwert aller α des jeweiligen Prozesses und $\Delta\mu$ den Fehler des Mittelwertes. σ bezeichnet die Standardabweichung und $\Delta\sigma$ deren Fehler. Die Abkürzungen in der Spalte Bereich bezeichnen den Bereich der Zelle, der betrachtet wird. Im einzelnen sind dies: gl (global) steht für die gesamte Zelle, hi (horizontal in) steht für innerhalb des horizontalen Streifens, ho (horizontal out) steht für außerhalb des horizontalen Streifens, si (slope in) steht für innerhalb des schrägen Streifens und so (slope out) steht für außerhalb des schrägen Streifens.

Eine übersichtliche Darstellung dieser Tabelle liefert Abb. 2.17. Dort wurden alle Werte der gemittelten α's getrennt nach Bereich, Vorzeichen und Mittendefinition aufgetragen. Dort ist deutlich zu erkennen, dass die berechneten Werte alle etwa den gleichen Wertebereich überschreiten, insbesondere auch unabhängig von der Stärke des angelegten Feldes. Eine genauere Betrachtung liefert ebenfalls keine Beziehung zwischen Vorzeichen der mitt-

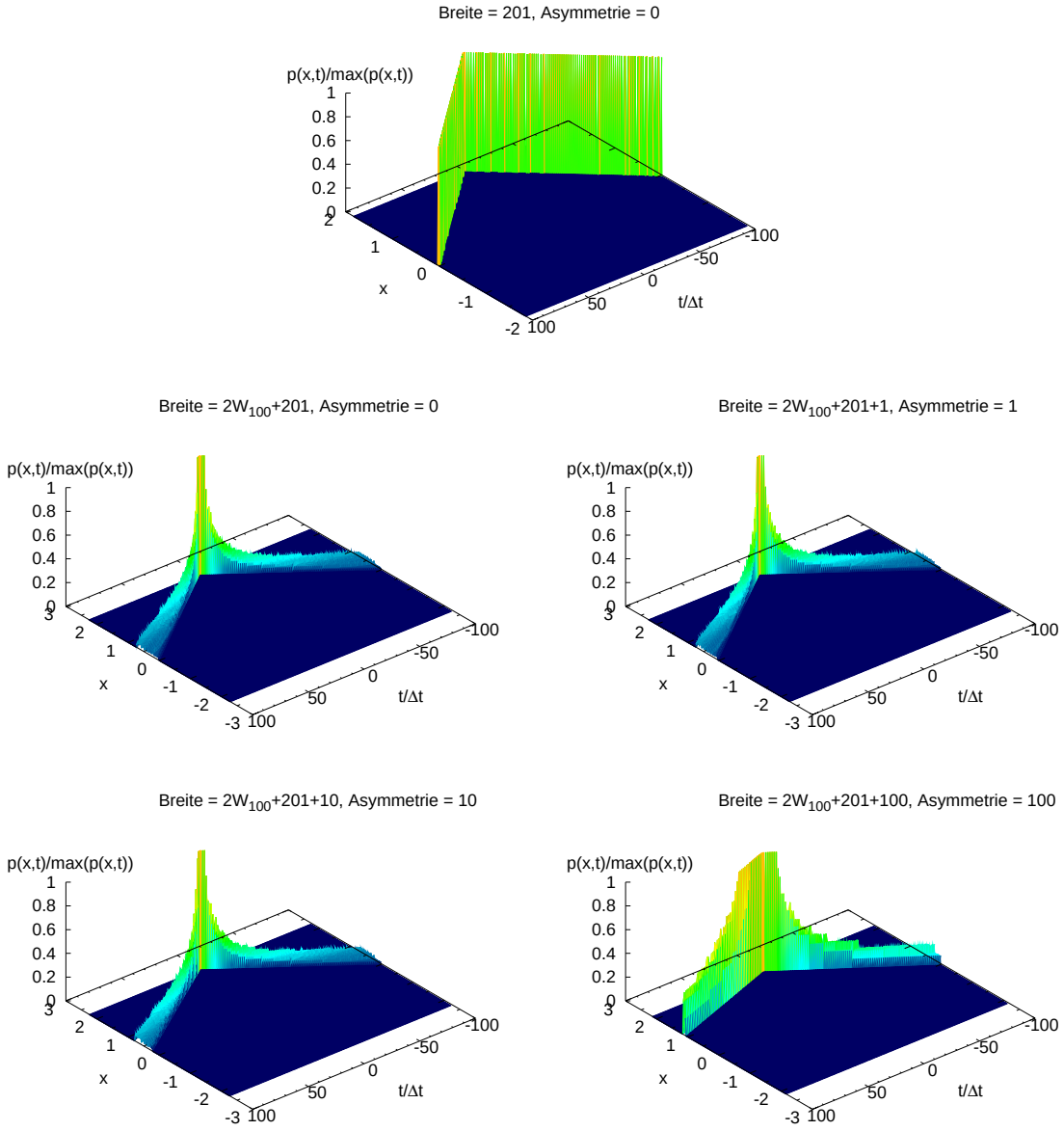


Abbildung 2.13: Diese Abbildung zeigt zu jedem Prozess das zweidimensionale Histogramm welches sich aus den positiven Fluktuationen ergibt. Dabei wurde jede Fluktuation symmetrisch um das Maximum zwischen den beiden Schwellendurchgängen ausgeschnitten und zum Histogramm hinzugefügt. Die entstehenden Strukturen erinnern zumeist an ein Bergmassiv mit sich weitenden Ausläufern und einem markanten Gipfel. Der Gipfel folgt direkt aus der Art der Datenaufnahme in das Histogramm. In diesem Beispiel ist die Mitte der Fluktuationen definiert als das Maximum zwischen den Schwellendurchgängen. Die Mitten sind mittig in der Horizontalen des Histogramms platziert. Demnach überlagern sich die Fluktuationen dort natürlich auch am stärksten. Ein Peak entsteht. Die Ausweitung der Ausläufer resultiert aus der größer werdenden Abweichung der Fluktuationen untereinander bei größer werdendem Abstand zum Maximum.

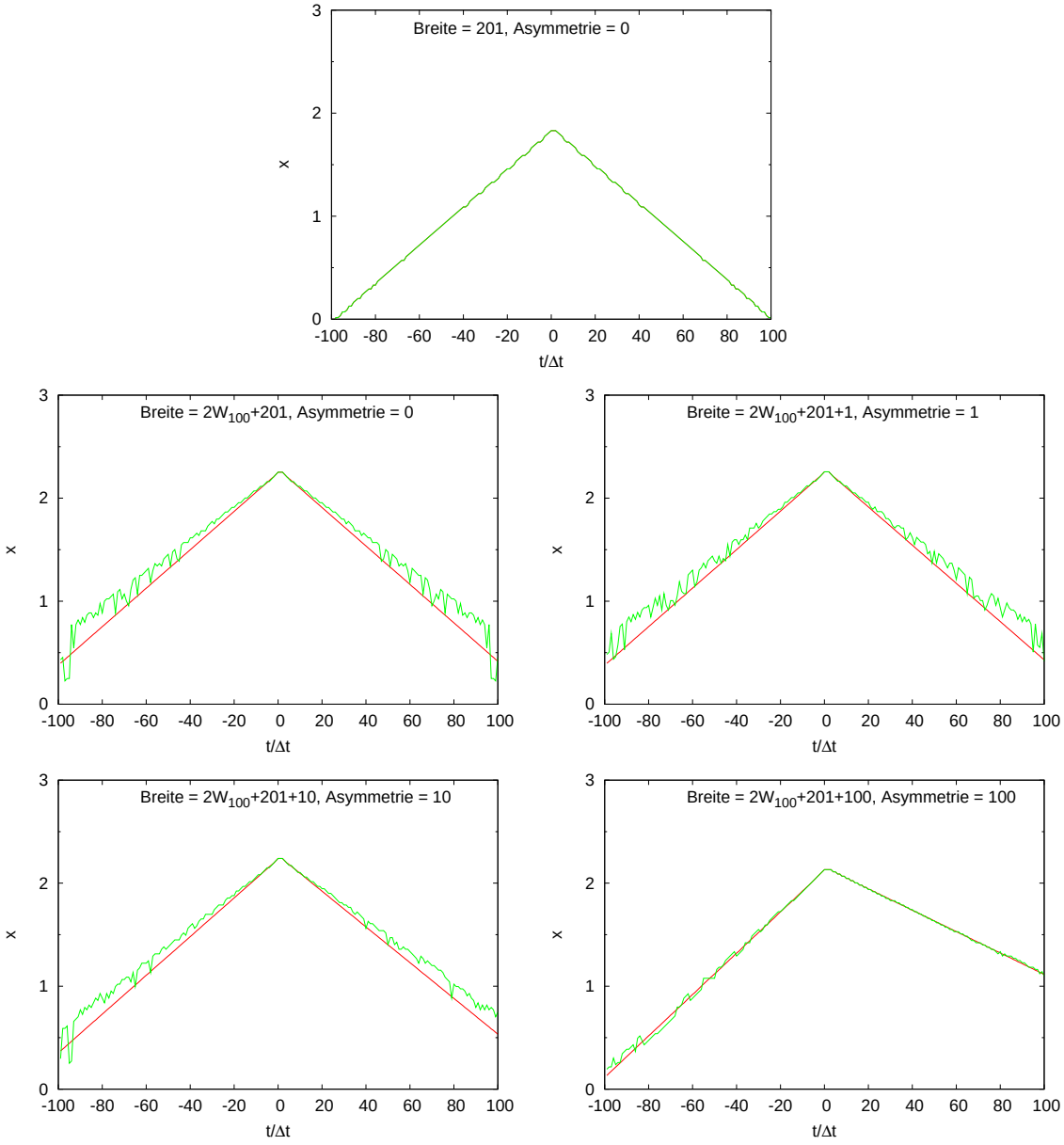


Abbildung 2.14: Diese Abbildung zeigt zu jedem Zeitschritt das Maximum (grün) und den Mittelwert (rot) des zweidimensionalen Histogramms. Man kann sich das Maximum vorstellen als den Grat der in Abb. 2.13 gezeigten „Bergmassive“. Der Mittelwert entsteht, wenn man den Grat senkrecht zur Zeitachse, also entlang des selben Zeitschritts, schneidet und im Schnittprofil mittelt. Es ist erkennbar, dass das tatsächliche Maximum pro Zeitschritt in einer relativ stark verrauschten mittleren Fluktuation resultiert, zumindest wenn die tatsächlich ja sehr glatte Form der originalen Fluktuationen bedacht wird. Durch die Mittelung wird das Rauschen praktisch vollständig unterdrückt.

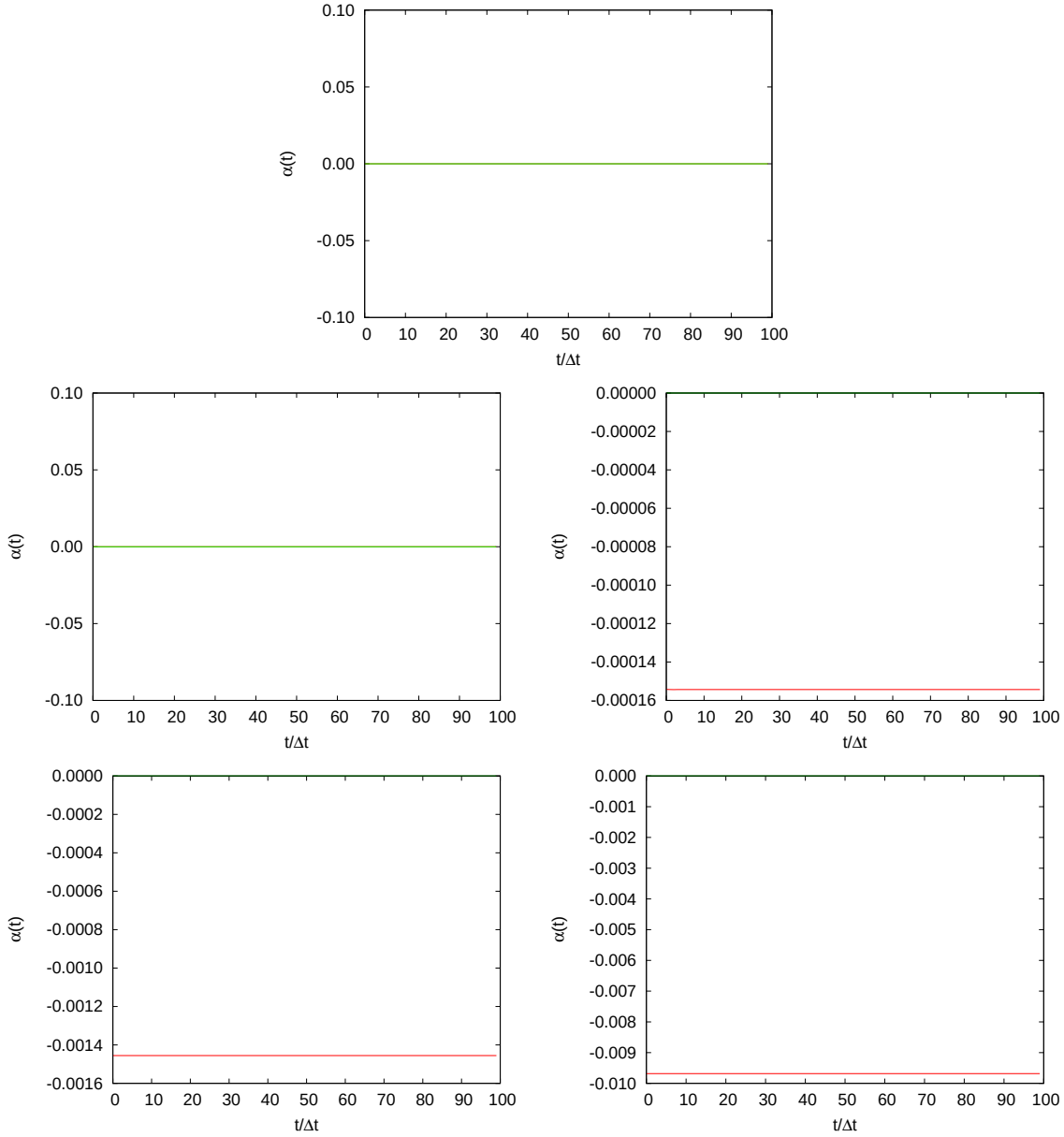


Abbildung 2.15: Diese Abbildung zeigt die Funktion $\alpha(i)$ für die Testsysteme. Die Funktion zeigt jeweils den Verlauf der gemittelten Differenz in den Steigungen der Flanken der Fluktuationen nach Abstand von deren Mittelpunkt. Die beiden ersten Systeme (oben und Mitte links) waren ohne Asymmetrie konstruiert und zeigen demnach Nulllinien. Bei den weiteren Systemen ist mit größer werdender Asymmetrie ein betragsmäßig größer werdender Wert der waagerechten Gerade zu beobachten. Eine waagerechte Gerade ist in diesen Beispielen zu erwarten, da die Flanken der synthetischen Fluktuationen von Geraden gebildet werden.

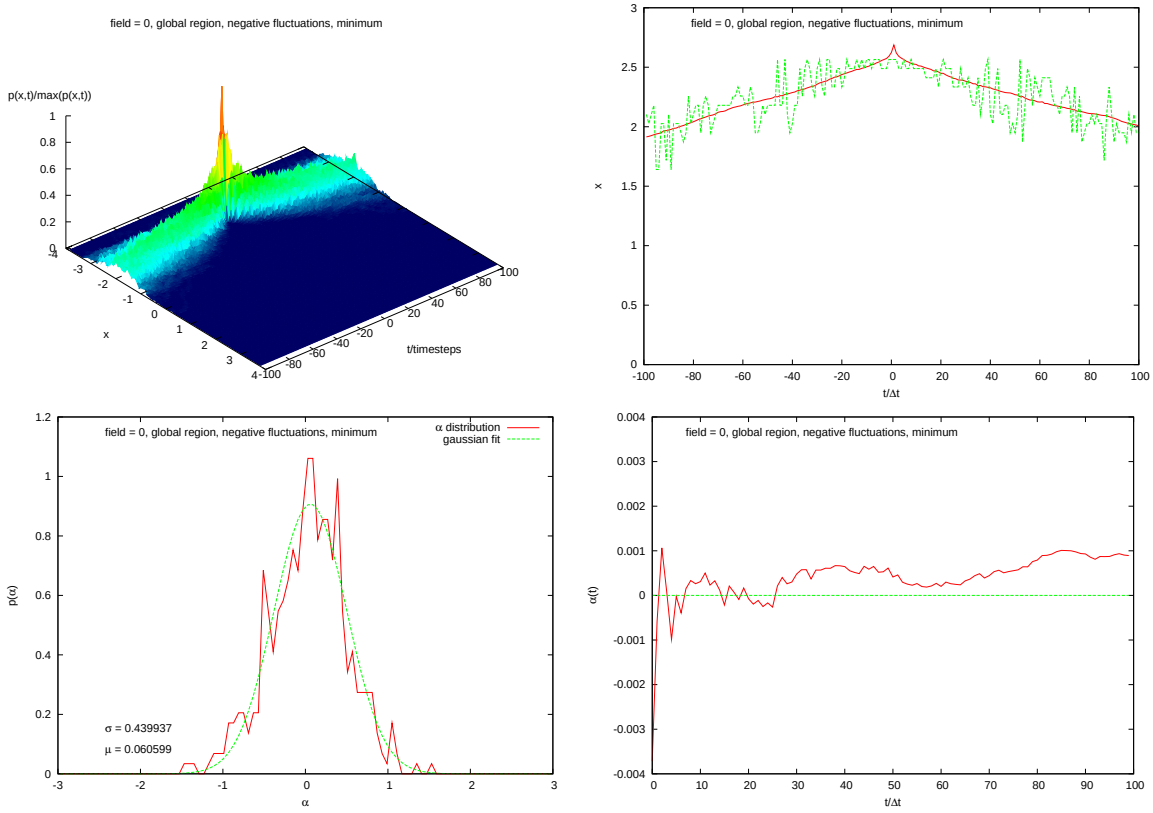


Abbildung 2.16: Diese Abbildung zeigt die für die Auswertung relevanten Darstellungen. Oben links ist das 2D-Stapelhistogramm aller negativen Fluktuationen zu sehen, welche im Prozess ohne äußeres Feld, im globalen Bereich der Zelle auftraten. Die Mitte der Fluktuationen war als das Minimum zwischen den Schrankendurchgängen definiert. Oben rechts ist der Verlauf des Maximums und der Mittelwerte entlang der Zeitachse des 2D-Histogramms zu sehen (siehe auch Abb. 2.14). Unten links ist das Histogramm $\alpha-$Werte der Fluktuationen zu sehen. Die grüne gestrichelte Linie zeigt einen Gauß-Fit, dessen Erwartungswert μ und Standardabweichung σ im Bild angegeben sind. Der Wert von μ entspricht dem Mittelwert aller $\alpha-$Werte des Prozesses. Unten rechts ist der Graph der Funktion $\alpha\,(i)$ für den untersuchten Prozess im Bereich $i \in [1, 100]$ zu sehen.
Es existiert je ein Satz solcher Bilder für jede Kombination aus Feldstärke, betrachtetem Bereich in der Elementarzelle, Vorzeichen der Fluktuationen, sowie der Mittendefinition. Ob der großen Zahl an Bildern, insgesamt also $100 \times 4 = 400$, die sich aus der Kombination ergibt, soll diese Abbildung lediglich als anschauliches Beispiel dienen. Eine vollständige Liste aller $\alpha-$Werte aus zeigt Tab. 2.1.

leren α's und den Vorzeichen der Fluktuationen. An dieser Stelle scheint die einzige mit den Daten vertretbare Aussage die Nichtexistenz von Asymmetrien in den betrachteten Prozessen zu sein.

Dennoch soll der Blick noch auf die Abhängigkeit der $\alpha-$Werte von der Fluktuationsbreite gerichtet werden. Sortiert nach unterschiedlichen Kriterien sind die jeweils einhundert Graphen von $\alpha\,(i)$ in den Abbn. 2.18, 2.19 und 2.20 dargestellt. Abb. 2.18 sortiert die Graphen nach der Kombination aus Vorzeichen der Fluktuationen und deren Mittendefinition. Da alle Graphen gleicher Feldstärke jeweils die gleiche Linienfarbe aufweisen, ist keine Unterscheidung nach Bereichen möglich. Dies ist durchaus gewollt, da einerseits an dieser Stelle eine weitere Differenzierung eher verwirrend als klärend wirken würde. Andererseits genügt auch ein Blick in diese Darstellung um erkennen zu können, dass es hier offenbar keinen Zusammenhang von gemittelten $\alpha-$Werten und Feldstärke zu geben scheint. Betrachtet man beispielsweise die Abbildung oben links, so kann man erkennen, dass alle Graphen für die Feldstärke $\epsilon = 1.0$ im Positiven liegen. Sollte dieses aber als Hinweis auf eine eventuelle Asymmetrie zu werten sein, so müsste diese insbesondere auch bei den höheren Feldstärken auftreten, was aber nicht der Fall ist. Hinzu kommt, dass in der Darstellung unten links die Graphen auch für die Feldstärke $\epsilon = 1.0$ wieder verteilt um die Null herum liegen, obwohl hier der einzige Unterschied der Darstellungen im Vorzeichen der Fluktuationen liegt. Derzeit ist aber noch kein Hinweis vorhanden, dass sich positive und negative Fluktuationen elementar verschieden verhalten sollten. Insgesamt liegen alle Graphen in etwa gleich um die Nullachse verteilt. Dies gilt insbesondere auch für die Graphen welche die Feldstärke $\epsilon = 0.0$ repräsentieren, und gerade diese zeigen in einigen der Bilder die Graphen mit dem größten Abstand zur Null.

Die Abbn. 2.19 und 2.20 bestätigen lediglich das bisher Geschlossene. Eine neue Erkenntnis gibt es dennoch. In den genannten Abbildungen findet die Sortierung der einzelnen Bilder nach Bereich bzw. der Feldstärke statt, und die Graphen innerhalb derer jeweils umgekehrt. Außerdem wird in beiden Abbildungen noch nach Vorzeichen unterschieden. Keine Unterscheidung machen beide Abbildungen, was die Mittendefinition der Fluktuationen angeht. Im Resultat kommen aber jeweils gleiche Graphendarstellungen doppelt und nahezu parallel vor. An dieser Stelle kann geschlossen werden, dass die Mittendefinition offenbar keinen erkennbaren Einfluss auf die Ergebnisse dieser Analyse hat.

Zusammenfassend soll an dieser Stelle bemerkt werden, dass die vorliegende Analyse des Lorentzgases im Rahmen dieser Arbeit nicht mit der Annahme von Asymmetrien in den großen Fluktuationen ($> 2.5\sigma$) der Stromstärke verträglich ist.

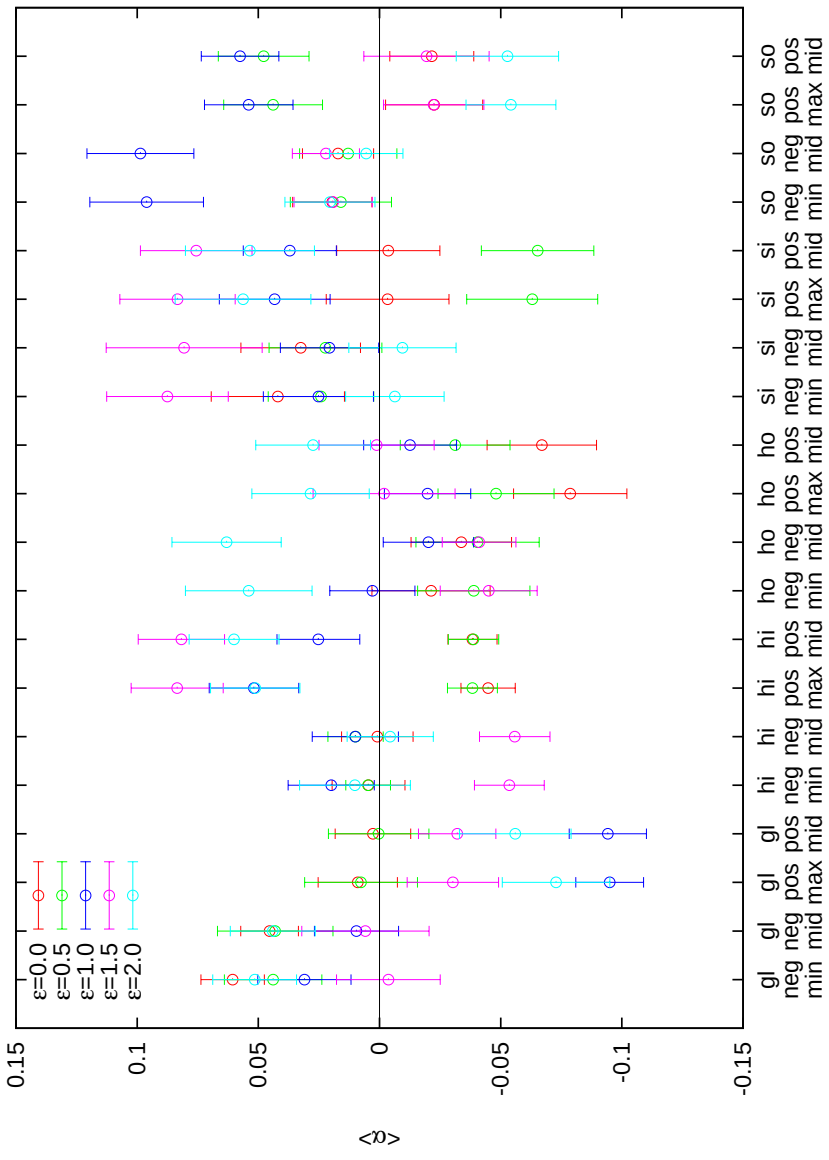


Abbildung 2.17: Diese Abbildung zeigt alle mittleren $\alpha-$Werte aus Tab. 2.1. Auf der waagerechten Achse sind die verschiedenen Kombinationen aus betrachtetem Bereich der Simulationszelle, Vorzeichen der Fluktuationen und deren Mittendefinition aufgetragen. Die verwendeten Abkürzungen stehen für: gl, globaler Bereich der Zelle; hi, im horizontalen Bereich; ho, außerhalb des horizontalen Bereichs; si, im schrägen Bereich; so, außerhalb des schrägen Bereichs; pos, positive Fluktuationen; neg, negative Fluktuationen; mid, Mitte der Fluktuationen entspricht der Mitte der Schwellendurchgänge; min und max, Mitte der Fluktuationen entspricht dem lokalen Minimum bzw. Maximum zwischen den Schwellendurchgängen. Zu jeder Kombination sind die Ergebnisse für alle betrachteten Werte des externen Feldes gezeigt.

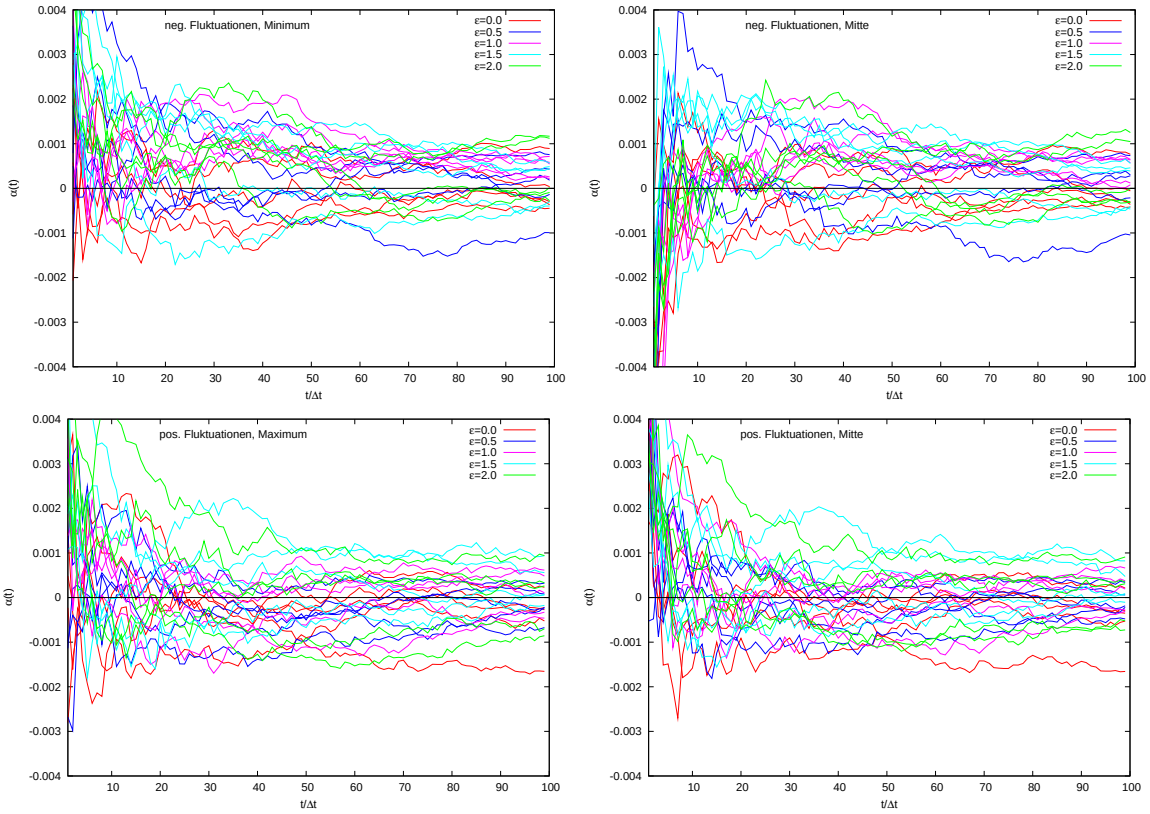


Abbildung 2.18: Diese Abbildung zeigt die Graphen der Funktion $\alpha\,(i)$ für die untersuchten Prozesse im Bereich $i \in [1, 100]$. Die Graphen innerhalb eines Bildes sind nach der Stärke des äußeren Feldes geordnet, und die vier Bilder zeigen jeweils eine Kombination aus Vorzeichen der Fluktuationen und deren Mittendefinition. Um welche Kombination es sich jeweils handelt ist in den Bildern angegeben. Aus der Abbildung geht nicht hervor, welche der fünf Funktionen einer Feldstärke für welchen Bereich der Simulationszelle steht. An dieser Stelle wurde aus Gründen der Übersichtlichkeit auf eine Unterscheidung verzichtet.

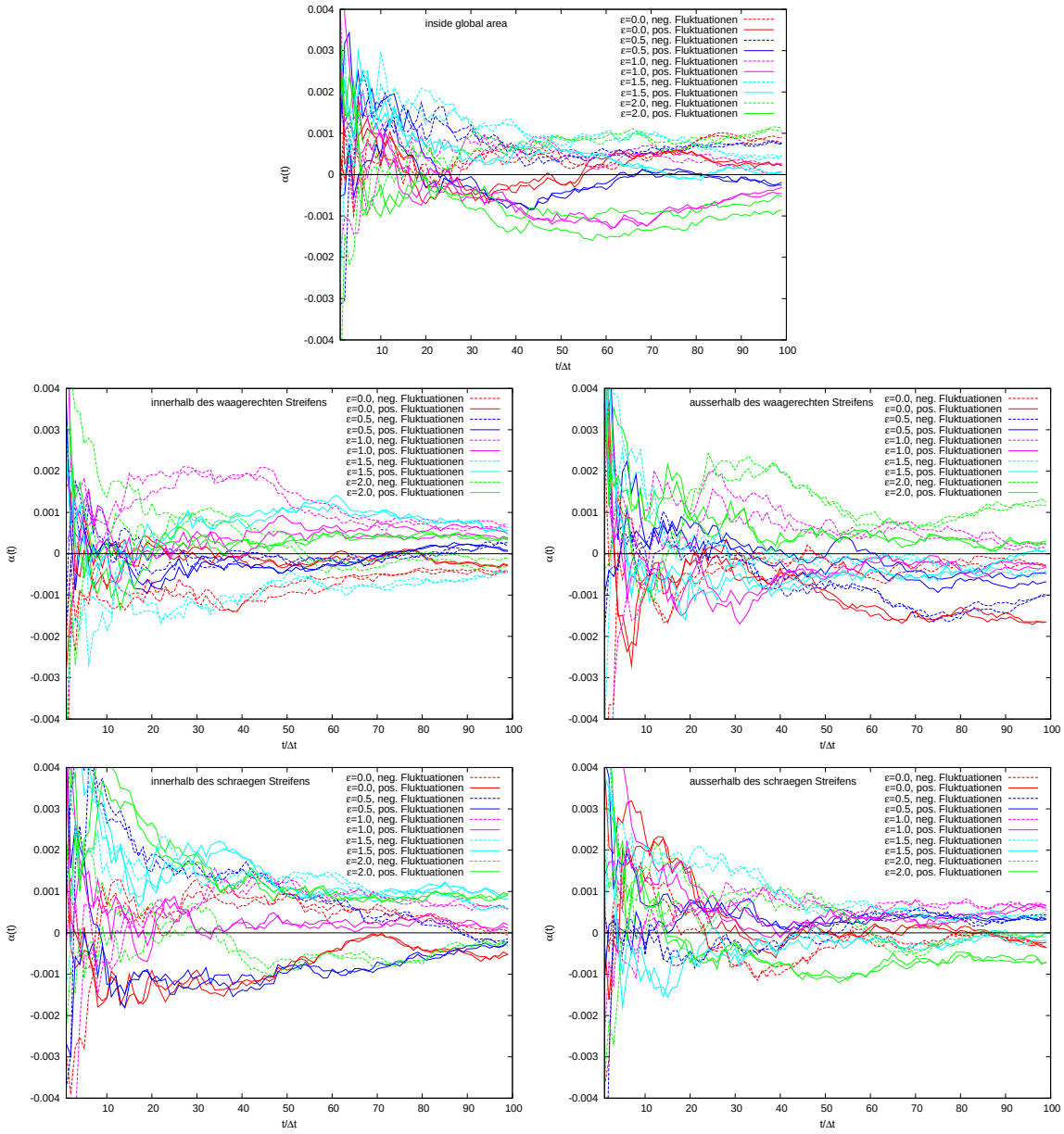


Abbildung 2.19: Diese Abbildung zeigt die Graphen der Funktion $\alpha(i)$ für die untersuchten Prozesse im Bereich $i \in [1, 100]$. Jedes der Bilder zeigt die Graphen, die zu einem bestimmten Bereich der Simulationszelle gehören. Der entsprechende Bereich ist im Bild angegeben. Zu jeder Feldstärke existieren vier Graphen. Die gestrichelten Linien repräsentieren die negativen Fluktuationen, und die durchgezogenen die positiven. Sowohl gestrichelte als auch durchgezogene Linien tauchen jeweils doppelt auf. Dahinter verbirgt sich die Unterscheidung der verschiedenen Mittendefinitionen. Aus Gründen der Übersichtlichkeit wurde hier aber darauf verzichtet, eigene Linienrepräsentationen zu wählen.

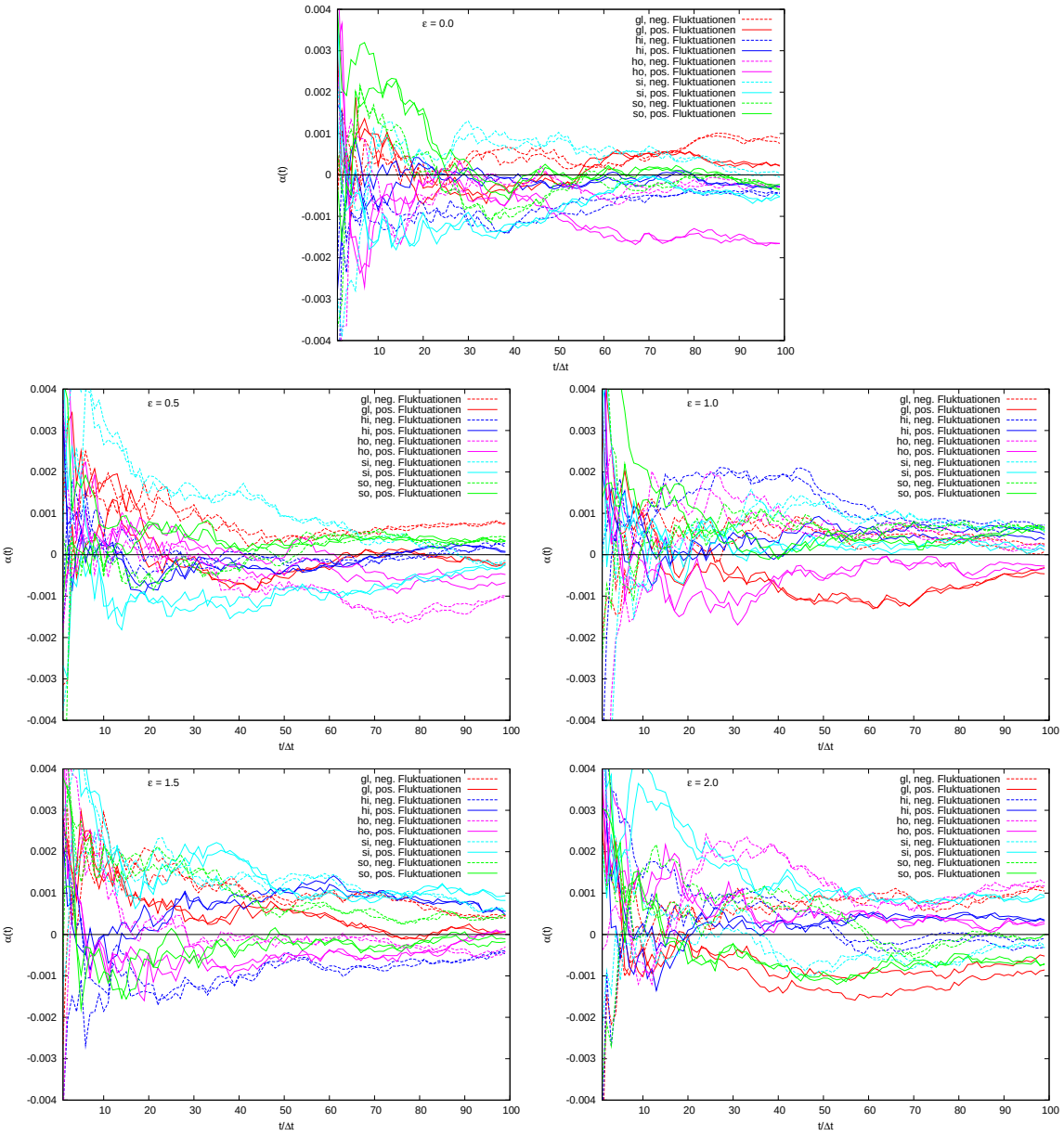


Abbildung 2.20: Diese Abbildung zeigt die Graphen der Funktion $\alpha(i)$ für die untersuchten Prozesse im Bereich $i \in [1, 100]$. Jedes der Bilder zeigt die zu einer bestimmten Stärke des äußeren Feldes gehörenden Graphen. Die entsprechende Feldstärke ist im jeweiligen Bild angegeben. Die gestrichelten Linien repräsentieren die negativen Fluktuationen und die durchgezogenen die positiven. Sowohl gestrichelte als auch durchgezogene Linien tauchen jeweils doppelt auf. Dahinter verbirgt sich die Unterscheidung der verschiedenen Mittendefinitionen. Aus Gründen der Übersichtlichkeit wurde hier aber darauf verzichtet eigene Linienrepräsentationen zu wählen.

Kapitel 3

Verteilungen von Energie und Geschwindigkeit in mikrokanonischen Gesamtheiten harter Kugeln

3.1 Motivation

Bekanntermaßen sind als Folge des Gleichverteilungssatzes in einem klassischen Fluid die Geschwindigkeitskomponenten normalverteilt mit Mittelwert $\mu = 0$ und Standardabweichung $\sigma = \sqrt{\frac{k_\mathrm{B}T}{m}}$ (Maxwell-Boltzmann-Verteilung). Daraus folgt durch einen Variablenwechsel in Kugelkoordinaten mit anschließender Integration über die Winkel, dass die Geschwindigkeitsbeträge der Maxwell-Verteilung folgen (manchmal wird auch diese Verteilung Maxwell-Boltzmann genannt). Eine zentrale Annahme für die Gültigkeit dieser Aussage ist eine genügend große Anzahl N an Molekülen, eigentlich der thermodynamische Grenzfall $N \to \infty$. Diese Annahme ist für reale Systeme in aller Regel ob der großen Teilchenzahl selbst bei geringen Stoffmengen (Avogadrozahl $N_\mathrm{A} \approx 6 \times 10^{23}$) leicht zu erfüllen. Auch in Computersimulationen, die mit sehr viel weniger Teilchen auskommen müssen, aber N mindestens einige hundert beträgt, ist diese Annahme zur Genüge erfüllt. Werden hingegen Systeme mit sehr kleinen Teilchenzahlen betrachtet ($N = 1$ bis 10), so weichen die sich ergebenden Verteilungen stark von der Maxwell-Boltzmann'schen und daher auch von der Maxwell'schen ab. Zusätzlich betrachten wir die Verteilung der kinetischen Energien, welche wiederum durch den einfachen Variablenwechsel $E = \frac{1}{2}mv^2$ mit dem Geschwindigkeitsbetrag zusammenhängt. Diese Verteilungen hängen zusätzlich von der Dimension d des Raumes ab (typischerweise ist man an $d = 2$ oder 3 interessiert).

Die Untersuchungen innerhalb dieses Kapitels beschäftigen sich mit der Bestimmung und Validierung von übergeordneten und allgemeingültigen Verteilungsfunktionen für beliebige Werte von N und d, welche im Falle großer Teilchenzahlen in die bekannten Verteilungen übergehen.

3.2 Einleitung

Das Problem der Geschwindigkeitsverteilung in einem Gas harter Kugeln wurde von Maxwell in einer im Jahre 1860 publizierten Veröffentlichung diskutiert.[30] Maxwell setzte seinerzeit Unabhängigkeit der Richtungskomponenten voraus, sowie auch Rotationsinvarianz der gesamten Verteilung. Die einzige Verteilung, welche den funktionalen Zusammenhang

$$f_{\vec{v}}(x,y,z) = \Phi(x^2+y^2+z^2) = f_{v_1}(x) f_{v_2}(y) f_{v_3}(z) \tag{3.1}$$

erfüllt hat die Form

$$f_{v_\alpha}(x) = A\exp(-Bx^2), \tag{3.2}$$

mit $\alpha = 1,2,3$. Diese einfache heuristische Herleitung findet sich bis heute in Lehrbüchern zur statistischen Physik und zur physikalischen Chemie.[31]

Im Jahre 1867 erkannte Maxwell,[32] dass Gleichung (3.2) die stationäre Lösung für die Dynamik des Gases darstellen sollte, und führte ein Konzept ein, welches später von Boltzmann als *Stoßzahlansatz* bezeichnet wurde. Dieses führte zu detaillierten Untersuchungen molekularer Stöße und zu Gleichungen, deren stationäre Lösungen mit Maxwells ursprünglicher Verteilung zusammenfielen. Dieser Weg wurde von Boltzmann verfolgt, welcher in den Jahren von 1868 bis 1871 eine Serie von Publikationen zu diesem Thema veröffentlichte.[33–35] Auf der Grundlage des *Stoßzahlansatzes* konnte Boltzmann zeigen, dass Maxwells Verteilung stationär ist. Die Ergebnisse werden in einem Buch von Tolman[36] zusammengefasst, sowie auch in den ersten Kapiteln eines Buches von ter Haar.[37]

Tolmans Analyse klassischer binärer Kollisionen harter Kugeln führte zu Ratengleichungen, welche als Übergangswahrscheinlichkeiten einer Markov-Kette interpretiert werden können. Der interessierte Leser mag an dieser Stelle Kapitel V und die Diskussion um Gleichung (45.3) auf Seite 129 in Tolmans Buch[36] nachschlagen. Die Verbindung zu Markov-Ketten wurde explizit von Constantini und Garibaldi[38,39] nachgewiesen, welche ein Modell von Brillouin[40] benutzten. Vor Constantini und Garibaldi nahm Penrose an, dass die Markov'sche Hypothese die Benutzung üblicher Methoden der statistischen Mechanik rechtfertigen könne.[41] Nach unserer Interpretation von Penrose verhalten sich Systeme von vielen interagierenden Teilchen effektiv wie Markov-Ketten. Hinzu kommt, dass die Anzahl der möglichen Zustände einer solchen Kette zwar sehr groß, aber endlich ist und demnach lediglich die Verwendung der Theorie für finite Markov-Ketten angewendet werden kann. Das statistische Gleichgewicht ist dann erreicht, wenn die Zustände eines Systems der Gleichgewichtsverteilung für finite Markov-Ketten gehorchen. Ist die Kette ergodisch oder irreduzibel, so existiert die Gleichgewichtsverteilung. Diese ist einzig und fällt mit der stationären Verteilung zusammen. In der englischsprachigen Literatur wird dieser Standpunkt als *Markovianism* bezeichnet. Tatsächlich konnte in einer Ver-

öffentlichung unseres Arbeitskreises zur Ehrenfest-Urne gezeigt werden, dass mit einer Markov-Kette das Verhalten eines realistischen Fluidmodells angenähert werden kann.[42]

Im Rahmen dieser Arbeit wurden Systeme mit N glatten elastischen harten Kugeln in d Dimensionen studiert. Obwohl die Entwicklung des Systems tatsächlich deterministisch verläuft, können die Geschwindigkeitskomponenten der Teilchen als Zufallsvariablen angesehen werden. Diese sollen nicht zur Vereinfachung diskretisiert, sondern als reelle Variablen angesehen werden.

Die theoretische Herleitung der folgenden Verteilungsfunktionen ist nicht Teil dieser Schrift. Dies ist darin begründet, dass die Herleitung noch nicht in allen Details zufriedenstellend abgeschlossen ist, noch Aufgabe des Autors war. Auf die Verteilungsfunktionen wird im Folgenden immer wieder eingegangen, so dass es dennoch an dieser Stelle unabdingbar erscheint diese anzugeben, auch wenn die zugehörige theoretische Behandlung hier nicht geliefert wird. Im einzelnen handelt es sich bei den Funktionen um die Verteilung der Geschwindigkeitskomponenten (3.3), die Verteilung der Geschwindigkeitsbeträge (3.5) und die Verteilung der Energien (3.7) von N glatten, elastischen Kugeln in einer Simulationsbox mit d Dimensionen.

Die Verteilung der Geschwindigkeitskomponenten $v_{i\alpha}$ der Teilchen $i = 1, \ldots, N$ in den Raumdimensionen $\alpha = 1, ..., d$, nimmt die Form

$$f_{v_{i\alpha}}(x) = \frac{\Theta\left(2ge - x^2\right)\Gamma\left(dg-1\right)}{2^{dg-2}\sqrt{2ge}\left(\Gamma\left(\frac{dg-1}{2}\right)\right)^2}\left(1 - \frac{x^2}{2ge}\right)^{\frac{dg-3}{2}} \tag{3.3}$$

an. Dabei bezeichnet g die Anzahl der freien Teilchen, $e = E/N = (1/N)\sum_{i=1}^{N} E_i$ die Energie pro Teilchen, Θ die Heaviside-Stufenfunktion und Γ die Gammafunktion. Eine Anmerkung zu der Anzahl der freien Teilchen g: Sei die tatsächliche Anzahl der Kugeln im System durch N bezeichnet, dann ist wegen möglicher Zwangsbedingungen $g \leq N$. Beispielsweise ist im Falle einer Box mit harten Wänden (HW) $g = N$, wenn keine Symmetrien bei Positionen und Geschwindigkeiten vorhanden sind, aber $g = N/2$ wenn alle Positionen und Geschwindigkeiten am Anfang punktsymmetrisch gewählt werden, wie wir später näher erläutern werden. Im Falle periodischer Randbedingungen (PR) ist $g = N-1$, wegen der Bedingung, dass der Gesamtimpuls erhalten bleibt. Für $N \to \infty$ konvergiert Gl. (3.3) zu einer Normalverteilung mit Mittelwert $\mu = 0$ und Varianz $\sigma^2 = \varepsilon d/2m$, d.h. zur bekannten Maxwell-Boltzmann-Verteilung

$$f_{v_{i\alpha}}(x) = \sqrt{\frac{m}{\pi\varepsilon d}}\exp\left(-\frac{mx^2}{\varepsilon d}\right). \tag{3.4}$$

Die Verteilung der Geschwindigkeitsbeträge v_i der einzelnen Teilchen $i = 1, \ldots, N$

ergibt sich zu

$$f_{v_i}(x) = \frac{x\Theta(x)\,\Theta\left(\sqrt{2g}-x\right)}{g\mathrm{B}\left(\frac{d}{2},\frac{d(g-1)}{2}\right)}\left(\frac{x^2}{2g}\right)^{\frac{d}{2}-1}\left(1-\frac{x^2}{2g}\right)^{\frac{d(g-1)}{2}-1}, \tag{3.5}$$

wobei B die Betafunktion bezeichnet. Diese ist über die Gammafunktion Γ definiert mit $\mathrm{B}(a,b) = \Gamma(a)\,\Gamma(b)\,/\Gamma(a+b)$. Für $N \to \infty$ konvergiert Gl. (3.5) zu einer Gammaverteilung

$$f_{v_i}(x) = \left(\frac{md}{\varepsilon}\right)^{d/2}\frac{(x/2)^{d-1}}{\Gamma(d/2)}\exp\left(-\frac{mdx^2}{4\varepsilon}\right). \tag{3.6}$$

Für $d = 3$ handelt es sich dabei um die bekannte Maxwell-Verteilung.

Die Verteilung der Energien E_i der einzelnen Teilchen $i = 1, \ldots, N$ lautet schließlich

$$f_{E_i}(x) = \frac{\Theta(x)\,\Theta(g-x)}{g\mathrm{B}\left(\frac{d}{2},\frac{d(g-1)}{2}\right)}\left(\frac{x}{g}\right)^{\frac{d}{2}-1}\left(1-\frac{x}{g}\right)^{\frac{d(g-1)}{2}-1}. \tag{3.7}$$

Für $N \to \infty$ konvergiert Gl. (3.7) wiederum zu einer Gammaverteilung

$$f_{E_i}(x) = \left(\frac{md}{2\varepsilon}\right)^{d/2}\frac{x^{d/2-1}\Theta(x)}{\Gamma(d/2)}\exp\left(-\frac{mdx}{2\varepsilon}\right), \tag{3.8}$$

welche für $d = 2$ die bekannte Boltzmann oder Boltzmann-Gibbs-Verteilung ergibt.

3.3 Aufbau der Simulation

Es wurden verschiedene Systeme harter Kugeln nach MD simuliert, und die Verteilungen der Geschwindigkeitsbeträge, der Geschwindigkeitskomponenten und der Energien der Teilchen ermittelt. Die einzelnen Systeme unterscheiden sich zum einen in der Anzahl der Teilchen und deren Startpositionen sowie Startgeschwindigkeiten und zum anderen in der Dimensionalität und den Randbedingungen der Simulationsbox.

Die Berechnungen sollen primär für sehr niedrige Teilchenzahlen ($N \leq 10$) erfolgen, aber auch für einige große Werte, um die Annäherung an die Maxwell-Boltzmann-Verteilung für die Geschwindigkeitsbeträge sicherzustellen. Konkret wurden Systeme mit den Teilchenzahlen $N = 2$, 3, 4, 10, 100, 1 000 und 10 000 simuliert. Es hat sich in der Anwendung gezeigt, dass spätestens ab 100 Teilchen keine Unterschiede in den Verteilungen mehr erkennbar sind. Deshalb werden im Weiteren die Systeme mit 10 000 Teilchen eine untergeordnete Rolle spielen. Dieser Sachverhalt ist in Abb. 3.1 dargestellt. Unterschiede in den Startpositionen umfassen unter anderem zufällige oder punktsymmetrische Anordnungen um den Mittelpunkt der Simulationsbox. Die Startgeschwindigkeiten werden

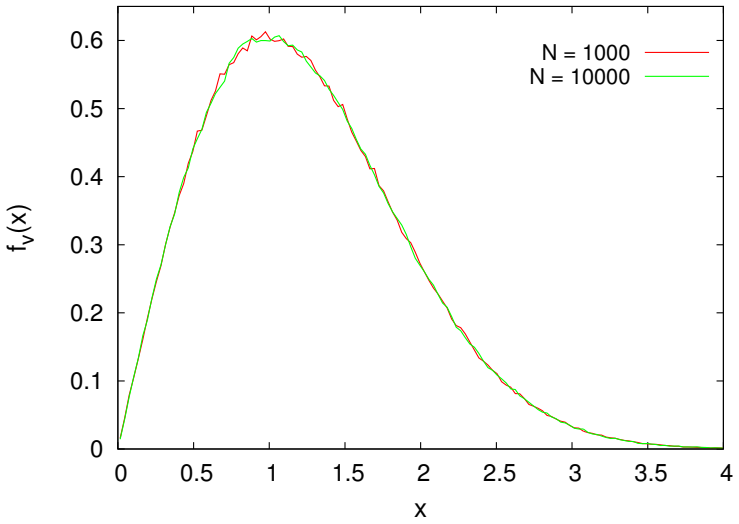


Abbildung 3.1: Diese Abbildung zeigt die Verteilung der Geschwindigkeitsbeträge im Vergleich zweier Systeme. Beide wurden in drei Dimensionen und periodischen Randbedingungen für eine Million Teilchenkollisionen simuliert. Der Unterschied der Systeme liegt in der Anzahl der harten Kugeln. Ein System beinhaltet eintausend und das andere zehntausend davon, dennoch sind beide Verteilungen praktisch deckungsgleich. Aus diesem Grund kann auf die Simulation von Systemen mit zehntausend Kugeln gänzlich verzichtet werden.

derart variiert, dass sich die Anordnung zu Beginn im Massenmittelpunktssystem befindet oder eben nicht. Zusätzlich werden noch Anordnungen erzeugt, die mit exakt punktsymmetrischen Geschwindigkeiten starten. Im Fall einer zweidimensionalen Simulationsbox werden die Teilchen mitunter auch als harte Scheiben bezeichnet.

Um einen Vergleich zu ermöglichen und eventuelle Fehler besser erkennen zu können, wurden zwei Programme entwickelt, welche die gleichen Systeme simulieren können. Der Unterschied zwischen ihnen besteht darin, dass eines der Programme eine sehr einfach gehaltene Version ohne jegliche Optimierung ist, während das andere die in Abschnitt 1.2 erwähnten Methoden umsetzt. Der Unterschied in der Berechnungsdauer der großen Systeme ist beachtlich. Während das einfache Programm für die Berechnung von einer Million Kollisionen in einem System mit 10 000 Kugeln viele Monate benötigen würde, erledigt das optimierte Programm diese Aufgabe in unter einer Stunde. Eine Gegenüberstellung von Benchmarkergebnissen zu verschiedenen Systemgrößen ist in Tab. 3.1 gegeben, und graphisch in den Abbn. 3.2 und 3.3. Entsprechend dieser Ergebnisse wurde nach einer ausgiebigen Test- und Vergleichsphase nahezu ausschließlich auf den optimierten Programmcode zurückgegriffen.

Parallel hierzu wurde ein MC-Simulationsprogramm entwickelt, welches Systeme der selben Art simuliert. Im Unterschied zu den zuvor beschriebenen Programmen, welche nach der ereignisgesteuerten MD die Teilchentrajektorien deterministisch berechnen, wird in diesem den Teilchen lediglich Geschwindigkeiten, aber keine Positionen zugeteilt. Der Algorithmus wählt nun in jeden Schritt zwei Teilchen zufällig aus und lässt sie unter einem

N	$t_{\mathrm{CPU}}^{\mathrm{opt}}/10^5$Kollisionen/s	$t_{\mathrm{CPU}}/10^5$Kollisionen/s
2	1,20	0,25
3	0,97	0,88
4	0,87	1,33
8	0,77	3,74
64	2,92	214,15
125	5,74	1 240,00
1 000	23,00	59 000 *
2 744	46,38	325 570 *
4 913	89,20	996 680 *
6 859	122,80	2 999 430 *
10 648	179,64	6 627 000 *

Tabelle 3.1: Diese Tabelle zeigt eine Gegenüberstellung der Benchmarkergebnisse für die beiden entwickelten MD-Simulationsprogramme. Es zeigt sich klar, dass für sehr geringe N keinerlei Geschwindigkeitsgewinn aus dem optimierten Programm hervorgeht und dieses ob des vorhandenen Mehraufwands sogar noch langsamer arbeitet. Bei größer werdenden Teilchenzahlen hingegen ist der Geschwindigkeitsgewinn enorm. Anm.: Die mit Stern gekennzeichneten Zeiten sind von 100 simulierten Kollisionen extrapoliert. Allen anderen Zeiten liegen tatsächlich 100 000 simulierte Kollisionen zugrunde.

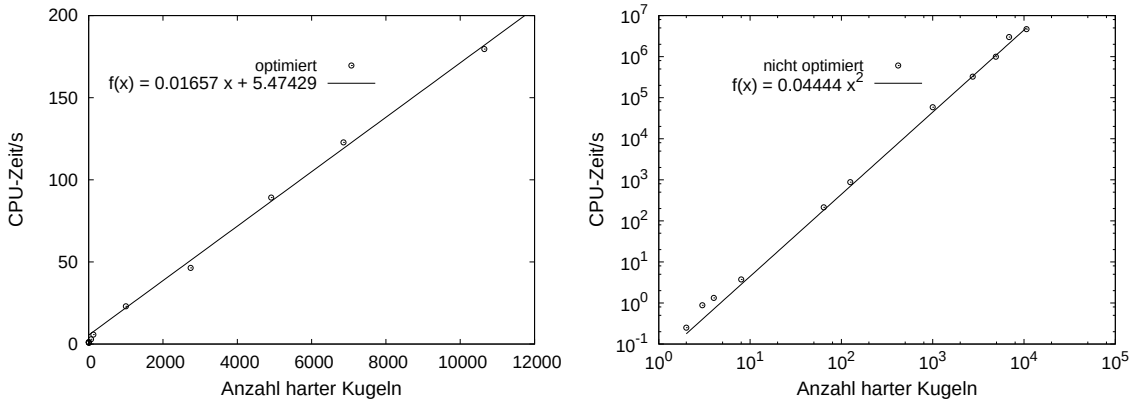


Abbildung 3.2: Diese Abbildung zeigt die Benchmarkergebnisse aus Tab. 3.1. Aufgetragen ist jeweils die benötigte Simulationszeit in Sekunden für 10^5 Kollisionen auf die Anzahl der simulierten Teilchen. Der linke Plot stellt die Ergebnisse für das optimierte Programm dar und es ergibt sich für Teilchenzahlen ab etwa eintausend ein grob linearer Zusammenhang. Die Funktion ist das Ergebnis eines linearen Fits. Der rechte Graph ist doppelt logarithmisch aufgetragen damit das Potenzgesetz sichtbar wird. Für diesen Plot ergibt sich $f(x)$ aus einem Fit mit einer einfachen quadratischen Funktion, welche den Zusammenhang auch recht gut wiedergibt.

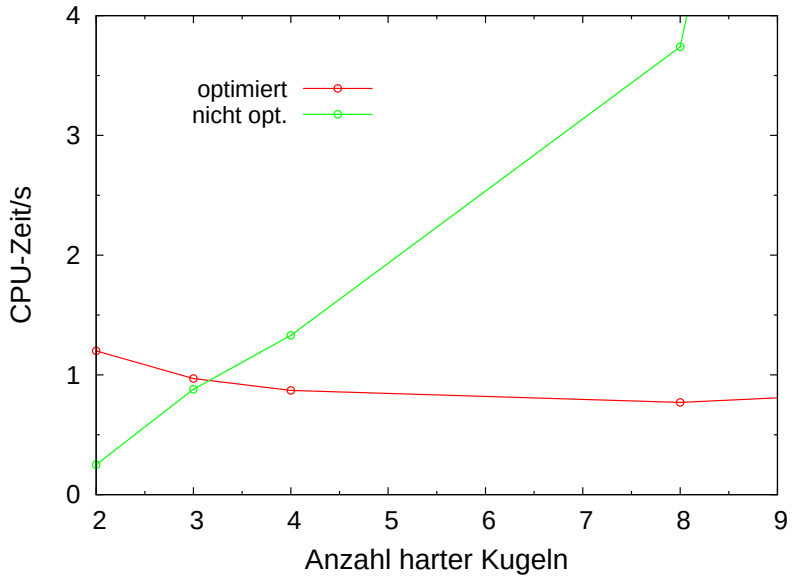


Abbildung 3.3: Diese Abbildung zeigt eine Gegenüberstellung der Benchmarkergebnisse aus Tab. 3.1 für die ersten vier Messwerte. Es ist zu erkennen, dass das optimierte Simulationsprogramm bei zwei oder drei Teilchen sogar schlechter abschneidet als das einfache. Diese Tatsache kommt allerdings nicht überraschend, denn die notwendigen Berechnungen und Unterteilungen, welche die Optimierung ausmachen, erzeugen auch einen zusätzlichen Rechenaufwand. Dieser macht sich aber schon bei vier Teilchen bezahlt.

zufälligen Winkel miteinander kollidieren, woraus sich neue Geschwindigkeiten für diese beiden Teilchen ergeben. Für HW werden, ebenfalls zufällig, Geschwindigkeitskomponenten invertiert, um Wandkollisionen darzustellen. Ein äquivalenter Ansatz zur zufälligen Ermittlung der Teilchenpaare ist es, immer wieder systematisch alle möglichen Paarungen zu durchlaufen, bis die geforderte Zahl an Kollisionen stattgefunden hat.[1]

Beide Ansätze, MD und MC, liefern in erster Näherung gleiche Ergebnisse. Bei genauerer Betrachtung sind jedoch kleine Unterschiede zu erkennen. Auf diese wird in Abschnitt 3.6 näher eingegangen.

Ein weiteres Instrument, um sich der korrekten Funktion der Simulationssoftware zu vergewissern, stellt die Graphikanwendung QMGA dar. Diese wurde ebenfalls im Rahmen dieser Arbeit entwickelt und wird in Kapitel 4 näher vorgestellt. An dieser Stelle genügt es zu bemerken, dass mit Hilfe von QMGA unter anderem Bildsequenzen aus den Teilchentrajektorien hergestellt und graphisch aufbereitet abgespielt werden können. Die Daten für diese Videos liefert in diesem Fall die Simulationssoftware selbst, indem periodisch die Teilchenpositionen auf Festplatte geschrieben werden. Auf diese Weise ist es möglich, die Teilchentrajektorien als bewegte Bilder zu verfolgen und visuell auf Konsistenz zu überprüfen.

3.4 Programminterna

Dieser Abschnitt soll den verwendeten Simulationscode etwas beleuchten. Das Design folgt einem vollständig objektorientierten Ansatz und ist auf einfache Erweiterbarkeit und Wiederverwendbarkeit des Codes ausgelegt. Die derzeit implementierten Klassen sind:

- Die Klasse *Cell* stellt eine Unterzelle der übergeordneten Simulationsbox dar. Sie beinhaltet unter Anderem die Funktionalität, um einer Zelle Teilchen zuzuweisen und diese wieder zu entfernen oder auch abzufragen, wie viele Teilchen sich in der Zelle befinden. Des weiteren können Verknüpfungen zu anderen Zellen hergestellt und abgefragt werden. Das bedeutet, eine Zelle kennt alle ihre nächsten Nachbarzellen, wenn die Verknüpfungen geeignet gesetzt wurden. Bei diesen Verknüpfungen handelt es sich um herkömmliche C++-Zeiger auf andere *Cell*-Objekte. Im Falle einer zweidimensionalen periodischen Box, welche in neun Unterzellen aufgeteilt werden soll, genügt es neun Zellen anzulegen und diese gegenseitig derart zu verknüpfen, wie es in Abb. 3.4 zu sehen ist. Für den zweidimensionalen Fall muss jede Zelle demnach über acht Zeiger auf ihre Nachbarzellen verfügen, während in drei Dimensionen schon sechsundzwanzig notwendig sind. Dieser Tatsache Rechnung tragend verfügt die *Cell*-Klasse über sechsundzwanzig vordefinierte Zeiger und die notwendigen Zugriffsfunktionen, um diese zu verwenden. Ob der großen Menge an Einzelzuweisungen wird diese Aufgabe von einer weiteren Klasse namens *Torus* erledigt.

- Die Klasse *Torus* übernimmt die Erstellung der Unterzellen. Sie kennt die Kantenlänge der Simulationsbox und die gewünschte ungefähre Kantenlänge der Unterzellen in den verschiedenen Raumrichtungen. Aus diesen Informationen werden die tatsächlichen Kantenlängen der Zellen und deren Anzahl bestimmt. Sie setzt die notwendigen Verknüpfungen, um gegebenenfalls einen dreidimensionalen Torus vollständig abzubilden. Sollte der periodische Übergang nicht erwünscht sein, so wird dieser im Laufe des Programms durch eine harte Kollision an einer Wand verhindert.

- Alle Ereignisse, die während des Simulationslaufs auftreten können, sind in der Klasse *Event* zusammengefasst. Die derzeit möglichen Ereignisse sind die Kollision zweier Teilchen, der Übergang eines Teilchens in eine Nachbarzelle, der Stoß eines Teilchens an einer harten Wand und eine Anweisung Daten auf die Festplatte zu schreiben. Die Wandstöße teilen sich auf in zwei Gruppen, einerseits in Stöße mit den Wänden der kubischen Simulationsbox und andererseits in Stöße mit harten zylindrischen Randbedingungen. Letztere können nur dann auftreten, wenn vor Programmstart entsprechende Eingaben vorgenommen werden. Die zylindrischen Randbedingungen lösen in diesem Fall die kubischen ab.

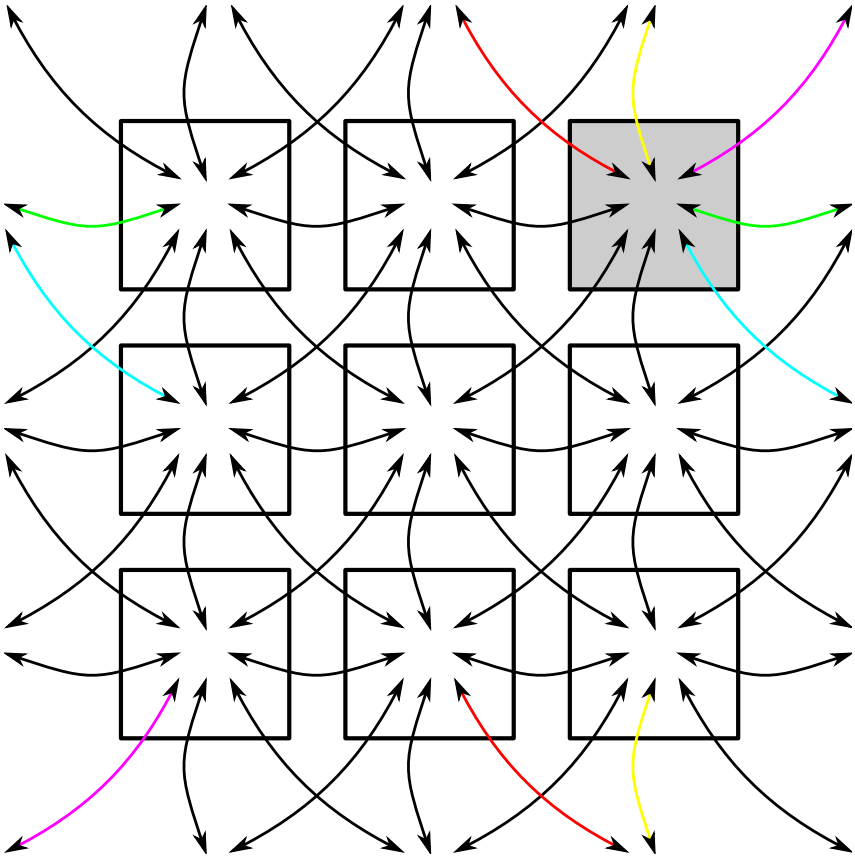


Abbildung 3.4: Diese Abbildung zeigt die Verknüpfungen, die für neun Zellen notwendig sind, um einen zweidimensionalen Torus zu modellieren. Die periodischen Verknüpfungen der grau hinterlegten Zelle oben rechts sind zum besseren Verständnis farbig gestaltet. Es ist zu erkennen, das in der zweidimensionalen Anordnung jede Zelle über acht Verknüpfungen verfügen muss, um alle nächsten Nachbarn anzubinden. Für eine dreidimensionale Anordnung erhöht sich diese Zahl auf sechsundzwanzig.

Ein Ereignis beinhaltet immer den Zeitpunkt zu dem es eintritt, wenn es nicht vorher invalidiert wird, was durch ein früher stattfindendes Ereignis geschehen kann. Weiterhin sind die null bis zwei beteiligten Teilchen angegeben, das heißt keins für einen Schreibevent, eins für einen Wandstoß oder einen Zellwechsel, und zwei für eine Teilchenkollision. Ebenso ist die Richtung eines eventuellen Wandstoßes beziehungsweise eines Zellwechsels angegeben.

- Die Klasse *Event_Schedule* bildet den Ereigniskalender. Hier werden die anfallenden Ereignisse gesammelt und nach Zeitpunkt des Eintretens in einem binären Baum[16] angeordnet. Für die Implementierung des Baumes wurde auf die Klasse *Multimap* der C++ Standard Template Library (STL) zurückgegriffen.[17]

- Die Klasse *Molecule* modelliert ein Teilchen welches, anders als der Name vermuten lassen könnte, eine harte Kugel darstellen soll. Die Daten umfassen die Position des Teilchens in der Simulationsbox und den Zellen, die aktuelle Geschwindigkeit und ein Etikett in Form einer fortlaufenden Nummerierung. Nach einem Stoß ist es notwendig, alle Ereignisse aus dem Kalender zu entfernen, an denen eines der Teilchen des aktuellen Stoßes beteiligt ist, da diese Ereignisse nicht mehr stattfinden werden. Um einen Mechanismus zur Verfügung zu stellen, der es ermöglicht, ohne längeres Suchen alle Ereignisse an dem ein Teilchen beteiligt ist zu entfernen, wird in jeder *Molecule*-Instanz protokolliert, an welchen Ereignissen dieses Teilchen teilnehmen wird. Das Protokoll wird in der Form von STL-*List* verketteten Listen geführt, welche aus Verknüpfungen auf die im Kalender gespeicherten *Event*-Objekte besteht. Bei den Verknüpfungen handelt es sich um die entsprechenden *Multimap*-Iteratoren des *Event_Schedule* Objektes. Die Bedeutung und Verwendung von Iteratoren kann ebenfalls im Buch von Josuttis zur STL[17] vertieft werden. An dieser Stelle sei lediglich angemerkt, dass es sich dabei um eine Art intelligenten Zeiger handelt.

- In der Klasse *Variables* werden die meisten Daten gesammelt, die für die Ausführung des Programms notwendig sind. Das betrifft sowohl die Eingabeparameter, als auch die vom Programm selbst berechneten Daten. Des weiteren befinden sich in dieser Klasse einige der Funktionen, welche zur Datenerhebung zuständig sind. Dies umfasst die Berechnung der Energieverteilung, der Geschwindigkeits- beziehungsweise Impulsverteilungen und der Teilchentrajektorien.

- Die Klasse *Hard_spheres* ist zuständig für alle Funktionalitäten die spezifisch die Dynamik der harten Kugeln betreffen. Das sind die Detektion der Stöße der Teilchen mit Wänden und untereinander, der Wechsel zwischen den Unterzellen und eventuellen verbotenen Teilchenüberlappungen, usw. In dieser Klasse wird die gesamte Arbeit verwaltet und verteilt. In einer Schleife werden so lange neue Ereignisse berechnet und ausgeführt, bis die gewünschte Zahl an Kollisionen im System erfolgt

Algorithm 3.1 Die main-Funktion des Simulationsprogramms ist sehr übersichtlich. Dies wird durch die Verlagerung des wirksamen Programmcodes in die Klassenstruktur ermöglicht. Ebenso ist eine unkomplizierte Erweiterbarkeit auf diese Weise sichergestellt, da relevante Teile des Codes durch eine Vererbung der Klassen ausgetauscht oder erweitert werden können. Ein Beispiel wäre die Implementierung eines alternativen Integrators um Teilchen mit ausgedehntem Potential zu behandeln. Die Entscheidung, welcher Integrator zum Einsatz kommt, kann bei geeigneter Implementierung zur Laufzeit des Programms erfolgen.

```
//----------------------------------------
//-------- main
//----------------------------------------
int main()
{
  // Read simulation control parametrs
  Hard_spheres integrator;
  integrator.set_variables( read_input( new Variables() ) );

  if( integrator.check_overlaps() == false )
    {
      cout << "--- Begin initialization of collisions." << endl;
      integrator.init_collision();
      cout << "--- Begin main loop." << endl;
      integrator.main_loop();
    }
  else
    {
      cerr << "ERROR: Initial configuration contains overlaps." << endl;
    }

  return(0);
}
```

ist.

Da die Arbeit vollständig innerhalb der Objekte erledigt wird, ergibt sich daraus eine außerordentlich übersichtliche main-Funktion. Ein Blick in den Programmcode in Alg. 3.1 macht dies deutlich. Dem gegenüber stehen nahezu 3 500 Zeilen Programmcode, innerhalb dessen die Berechnungen erfolgen. Wie eingangs schon erwähnt, war Erweiterbarkeit ein Designziel dieses Programms. Sollen anstatt harter Kugeln Teilchen mit weichem Potential simuliert werden, so wäre zum Beispiel die Implementierung eines alternativen Integrators leicht möglich. Auf die übrige Programmfunktionalität könnte in weiten Teilen zurückgegriffen werden. Es müsste nur zu Beginn des Programms, während der Laufzeit, in einer Abfrage eine Entscheidung zur Teilchensorte erfolgen.

3.5 Zusammenfassung der Ergebnisse

In diesem Abschnitt sollen die gesammelten Daten verglichen und in Bezug gesetzt werden, siehe Abbn. 3.5 bis 3.8. Die entsprechenden Verteilungen wurden jeweils auf drei

verschiedene Weisen bestimmt: Einmal durch Zeichnen der theoretischen Kurve, welche es zu überprüfen gilt, einmal durch MC-Simulation der Teilchen und ein weiteres mal durch ereignisgesteuerte MD-Simulation. Die Ergebnisse sind in erster Näherung konsistent, doch bei genauer Betrachtung werden auch Unterschiede offenbar. Auf einige der Abweichungen wird in den nächsten Abschnitten eingegangen.

Grundsätzlich gliedern sich die Ergebnisse wie folgt:

- Vier Gruppen: 2D periodisch, 2D mit Wänden, 3D periodisch und 3D mit Wänden
- Sechs Systeme je Gruppe: $N = 2$, $N = 3$, $N = 4$, $N = 10$, $N = 100$ und $N = 1\,000$
- Je System eine theoretische Kurve und zwei Simulationen, das sind MC und MD
- Je Simulation drei Verteilungen: Geschwindigkeitsbeträge, Energien und Geschwindigkeitskomponenten der Teilchen

Aus dieser Gliederung ergeben sich vier mal drei Abbildungen zu je 18 Graphen von denen sich jeweils drei gleichen sollten. Die MD-Simulationen, die für die Gesamtauswertung berechnet wurden, sind folgenden Regeln unterworfen:

1. Alle Simulationsboxen sind quadratisch im zweidimensionalen Fall und kubisch im dreidimensionalen.
2. Die Teilchendichte ist für alle zweidimensionalen Systeme und alle dreidimensionalen Systeme gleich. Aus programmtechnischen Gründen liegt die kleinste mögliche Boxlänge für das optimierte Programm bei drei. Die Dichte wurde nun so festgelegt, dass jeweils die Simulationsbox der Zweiersysteme eine Kantenlänge von drei aufweist.
3. Alle Systeme starten mit einem Gesamtimpuls von $\vec{P} = \vec{0}$.
4. Die mittlere Teilchenenergie beträgt bei allen Systemen $\epsilon = E/N = 1$, die Gesamtenergie demnach $E = N$.
5. Die Startsysteme mit gleichem N sind für Simulationen mit HW und PR gleich. Eine Ausnahme dazu bildet $N = 2$; hier muss für HW sichergestellt werden, dass der Gesamtimpuls zu Beginn nicht verschwindet. Der Grund hierzu ergibt sich aus den Beobachtungen in Kapitel 3.6.5.
6. Alle Simulationen starten mit einer Equilibrierungsphase von 10^5 Teilchen-Teilchen-Kollisionen.
7. Die eigentliche Simulationsdauer zur Datenaufnahme schließt sich mit weiteren 10^6 Kollisionen an die Equilibrierungsphase an.

Eine vollständige Liste aller simulierten Systeme, jeweils inklusive einer Abbildung der Startkonfiguration, zeigt Tab. 3.2. Bei diesen Systemen handelt es sich um jene, die für die Simulationen verwendet wurden, deren Ergebnisse in den Abbn. 3.5 bis 3.8 dargestellt sind. In der Betrachtung der Abweichungen in den nächsten Abschnitten kommen weitere Systeme zum Einsatz. Diese sind dann im jeweiligen Abschnitt dargestellt.

3.6 Auswertung

In diesem Kapitel soll eine Auswahl relevanter Simulationen und deren Auswertung gezeigt werden. Zunächst soll die Analyse anhand eines einfachen Systems vorgestellt werden, welches keine Abweichungen zum erwarteten beziehungsweise gewünschten Verhalten zeigt. Die im weiteren Verlauf betrachteten Systeme weichen zum Teil von den Erwartungen ab und müssen deshalb näher untersucht werden. Eine Beschäftigung mit einem unkomplizierten System ist daher zunächst angeraten. Wenn der Weg der Analyse klarer geworden ist, kann die Aufmerksamkeit vermehrt auf die Abweichungen gerichtet werden. Es werden nicht alle hier vorgestellten Abbildungen in jeder der folgenden Auswertungen erneut gezeigt. Im speziellen Fall kann auf einzelne Darstellungen verzichtet werden, da aus ihnen kein Informationsgewinn resultiert. An dieser Stelle soll aber die gesamte Sammlung der Analysemöglichkeiten betrachtet und vorgestellt werden.

Betrachtet wird nun ein System bestehend aus drei gleich großen, runden, zweidimensionalen harten Scheiben gleicher Masse in einer quadratischen, ebenfalls zweidimensionalen Simulationsbox mit harten Wänden. Eine graphische Darstellung des Systems zeigt Abb. 3.9. Dort ist zu erkennen, dass die Scheiben ungeordnet in der Box starten, wobei ein einzelnes Teilchen die gesamte Energie trägt. Dieses Teilchen bewegt sich mit der Geschwindigkeit $\vec{v} = \left(\sqrt{2E/m}, 0, 0\right)$, wobei m die Masse der einzelnen Teilchen mit $m = 1$ und die Gesamtenergie E mit $E = 3$ festgesetzt wird. Auch im weiteren Verlauf wird jeweils davon ausgegangen, dass alle Teilchen die gleiche Masse mit $m = 1$ besitzen. Ebenso wird die Gesamtenergie immer den Wert $E = N$ aufweisen, wodurch $e = E/N = 1$, wobei N die Anzahl der Teilchen im System angibt. Soweit nicht anders angegeben, wurden alle Simulationen für eine Dauer von einer Million Teilchen-Teilchen-Kollisionen berechnet.

Die Verteilungen, welche durch unsere Formeln (3.3, 3.5, 3.7) reproduziert werden sollen, sind in Abb. 3.10 zu sehen. Man kann erkennen, dass insbesondere die Verteilung der Geschwindigkeitsbeträge nicht der Maxwell-Boltzmann-Verteilung entspricht. Eine Zusammenfassung dieser Verteilungen für die Teilchenzahlen $N \in \{2, 3, 4, 10, 100, 1\,000\}$ zeigen folgende Abbildungen: Für eine zweidimensionale Simulationsbox Abb. 3.5 (harte Wände (HW)) und Abb. 3.7 (periodische Randbedingungen (PR)). Die Verteilungen für dreidimensionale Boxen zeigen Abb. 3.6 (HW) und 3.8 (PR).

Hilfsmittel, um das Verhalten der Systeme besser verstehen und kontrollieren zu können, stellen Verteilungen des Gesamtimpulses und seiner Komponenten dar, siehe

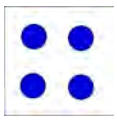
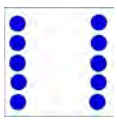
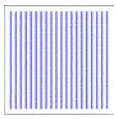
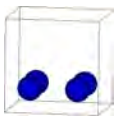
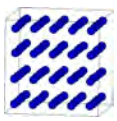
N	d	Boxlänge $L = \left[\sqrt[d]{N/\rho}\right]$	Dichte $\rho = [N/L^d]$	Start-konfiguration
2	2	3.0	$\frac{2}{9}$	
3	2	3.67423461	$\frac{2}{9}$	
4	2	4.24264069	$\frac{2}{9}$	
10	2	6.70820393	$\frac{2}{9}$	
100	2	21.21320344	$\frac{2}{9}$	
1 000	2	67.08203932	$\frac{2}{9}$	
2	3	3.0	$\frac{2}{27}$	
3	3	3.43414273	$\frac{2}{27}$	
4	3	3.77976315	$\frac{2}{27}$	
10	3	5.12992784	$\frac{2}{27}$	
100	3	11.05209450	$\frac{2}{27}$	
1 000	3	23.81101578	$\frac{2}{27}$	

Tabelle 3.2: Diese Tabelle fasst alle für die Auswertung simulierten Systeme in einer Übersicht zusammen. N steht für die Anzahl der Teilchen im System und d gibt die Dimensionalität an. In der Formel für die Dichte wurde Boxlänge mit *Boxl* abgekürzt. Bei der angegebenen Dichte handelt es sich um eine Teilchenzahldichte, also Zahl der Teilchen pro Volumen.

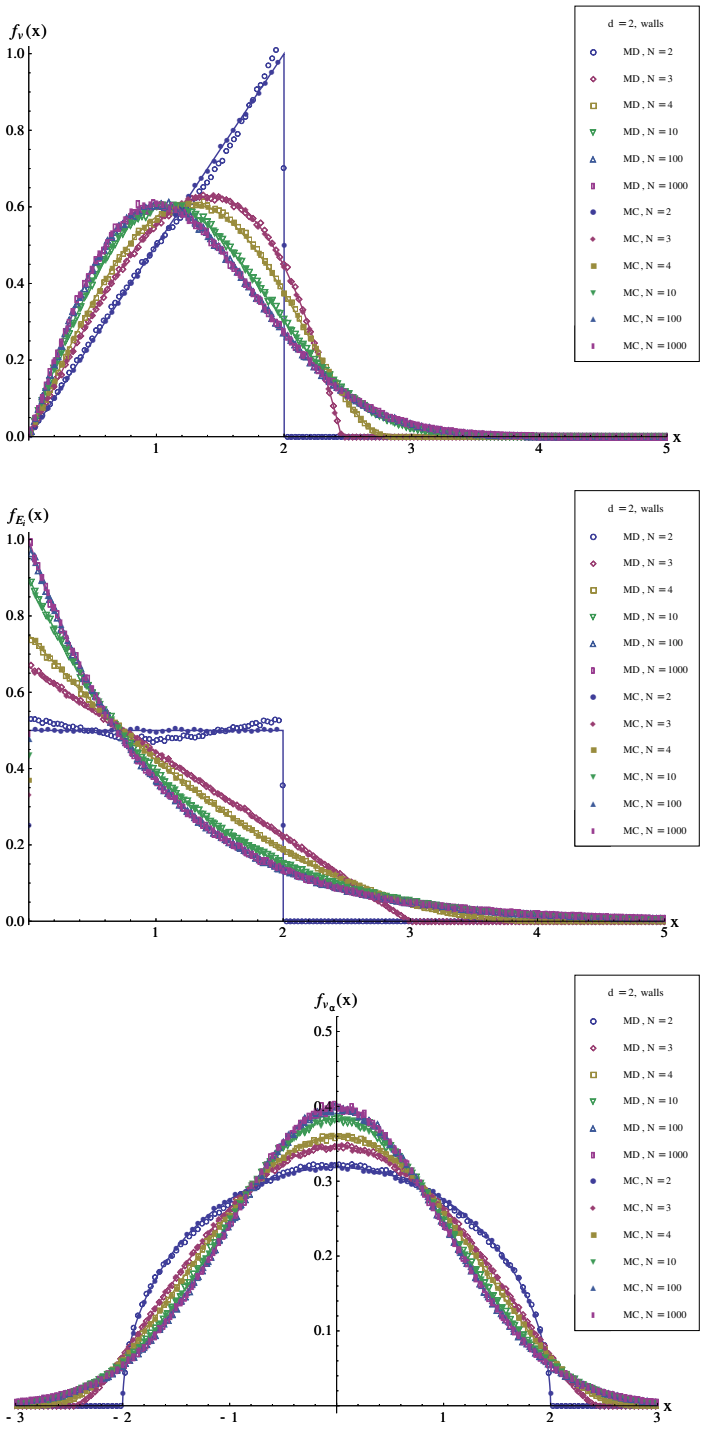


Abbildung 3.5: Diese Abbildung zeigt die Verteilungen für zweidimensionale Simulationsboxen mit harten Wänden. Oben ist die Verteilung der Geschwindigkeitsbeträge der harten Scheiben zu sehen. Die mittlere Abbildung zeigt die Verteilung der Teilchenenergien und die untere die Verteilung der Geschwindigkeitskomponenten. Auffällig ist einerseits die sehr gute generelle Übereinstimmung der Ergebnisse und andererseits die dennoch vorhandene Abweichung bei der Molekularsimulation im Fall $N = 2$, welche in Abschnitt 3.6.5 beleuchtet wird.

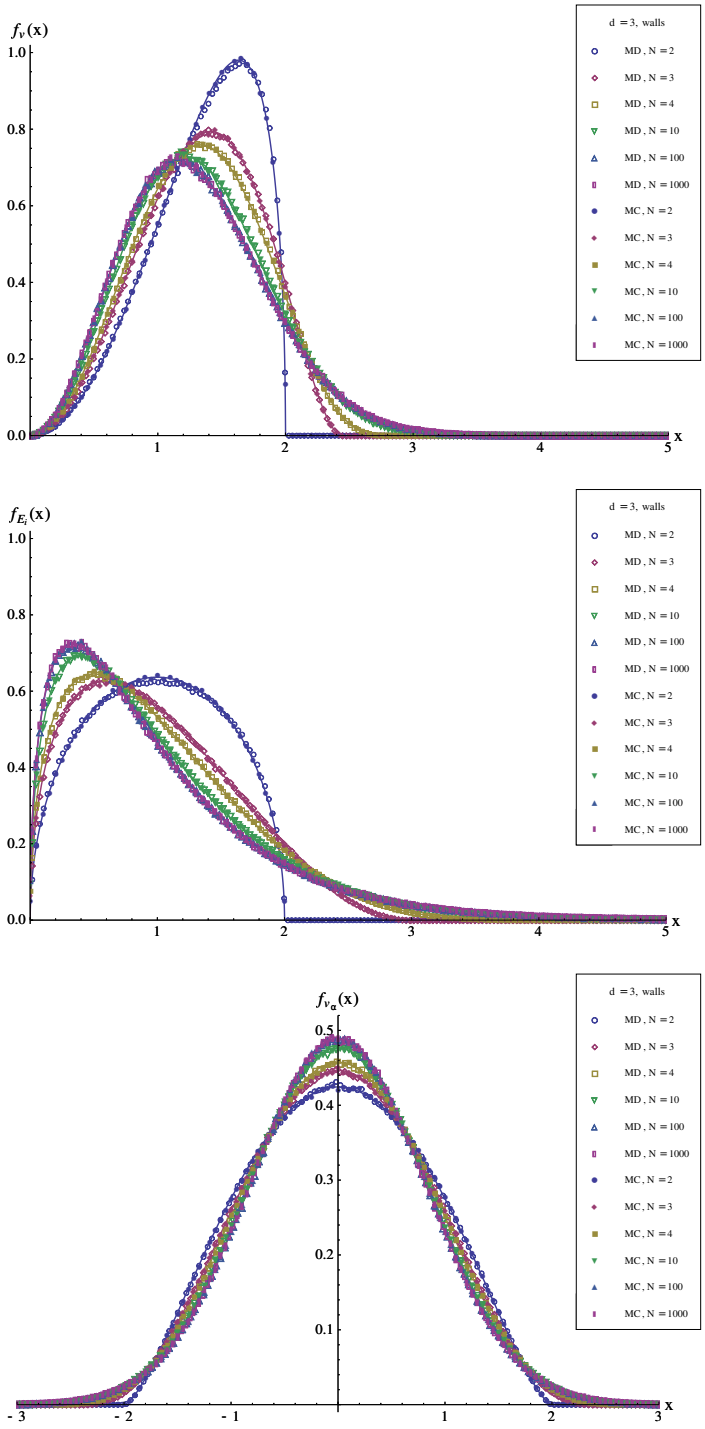


Abbildung 3.6: Diese Abbildung zeigt die Verteilungen für dreidimensionale Simulationsboxen mit harten Wänden. Oben ist die Verteilung der Geschwindigkeitsbeträge der harten Kugeln zu sehen. Die mittlere Abbildung zeigt die Verteilung der Teilchenenergien und die untere die Verteilung der Geschwindigkeitskomponenten. Wie auch für den zweidimensionalen Fall mit harten Wänden ist die generelle Übereinstimmung der Ergebnisse sehr gut, jedoch kommt es auch im dreidimensionalen Fall zu einer kleinen Abweichung bei der Molekulardynamikberechnung zweier harter Kugeln.

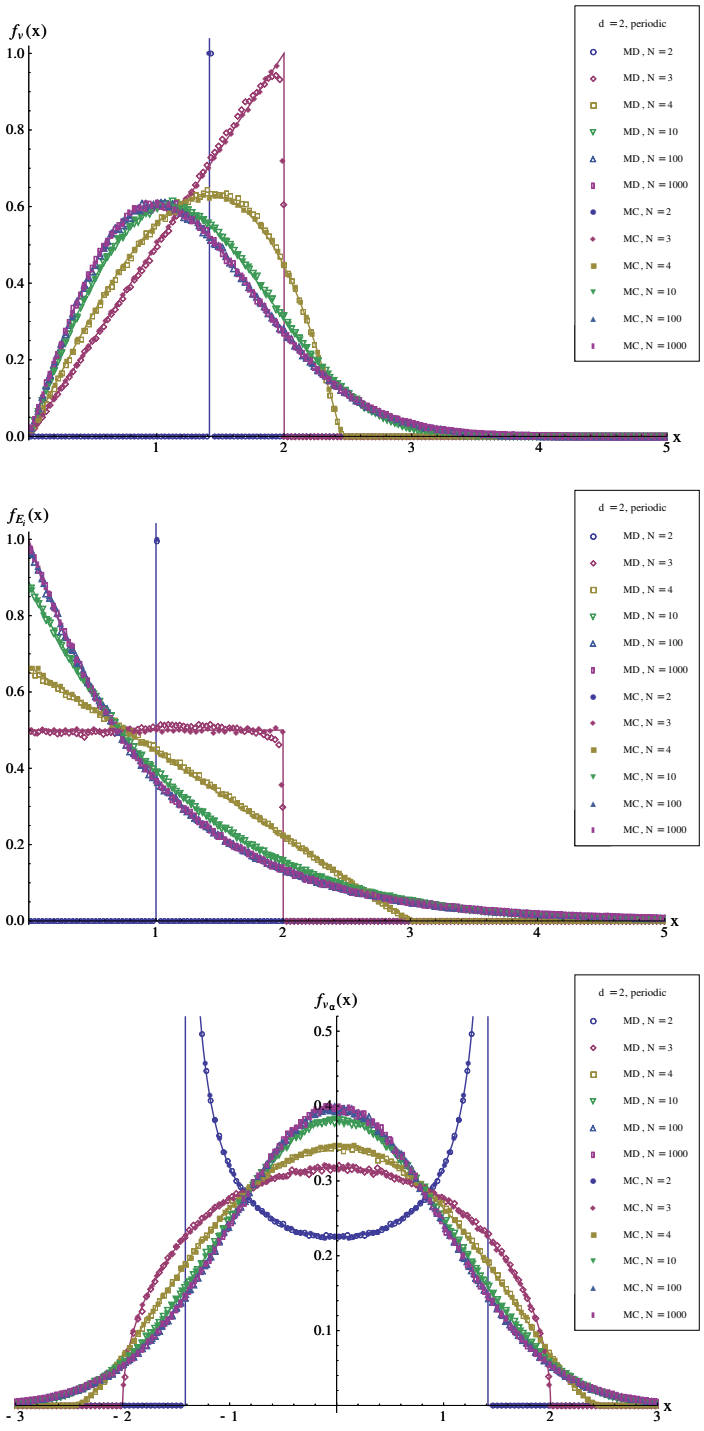


Abbildung 3.7: Diese Abbildung zeigt die Verteilungen für zweidimensionale Simulationsboxen mit periodischen Randbedingungen. Oben ist die Verteilung der Geschwindigkeitsbeträge der harten Scheiben zu sehen. Die mittlere Abbildung zeigt die Verteilung der Teilchenenergien und die untere die Verteilung der Geschwindigkeitskomponenten. Die Übereinstimmung der Ergebnisse ist auch in diesen Fällen sehr gut. Dennoch gibt es für den Fall periodischer Randbedingungen einen noch ungeklärten Effekt, der offensichtlich zu einer leichten Abweichung der Verteilungen für Energie und Geschwindigkeitsbeträge führt, aber die Verteilungen der Geschwindigkeitskomponenten nicht beeinflusst. Deutlich zu sehen ist diese Abweichung für $N = 3$ in den beiden oberen Abbildungen. Jedoch offenbart ein gründlicher Blick ebenfalls Abweichungen für $N = 4$, wobei der Effekt aber stark abnimmt. Für $N = 10$ ist er schon, auch bei starker Vergrößerung am Bildschirm, nicht mehr zu erkennen.

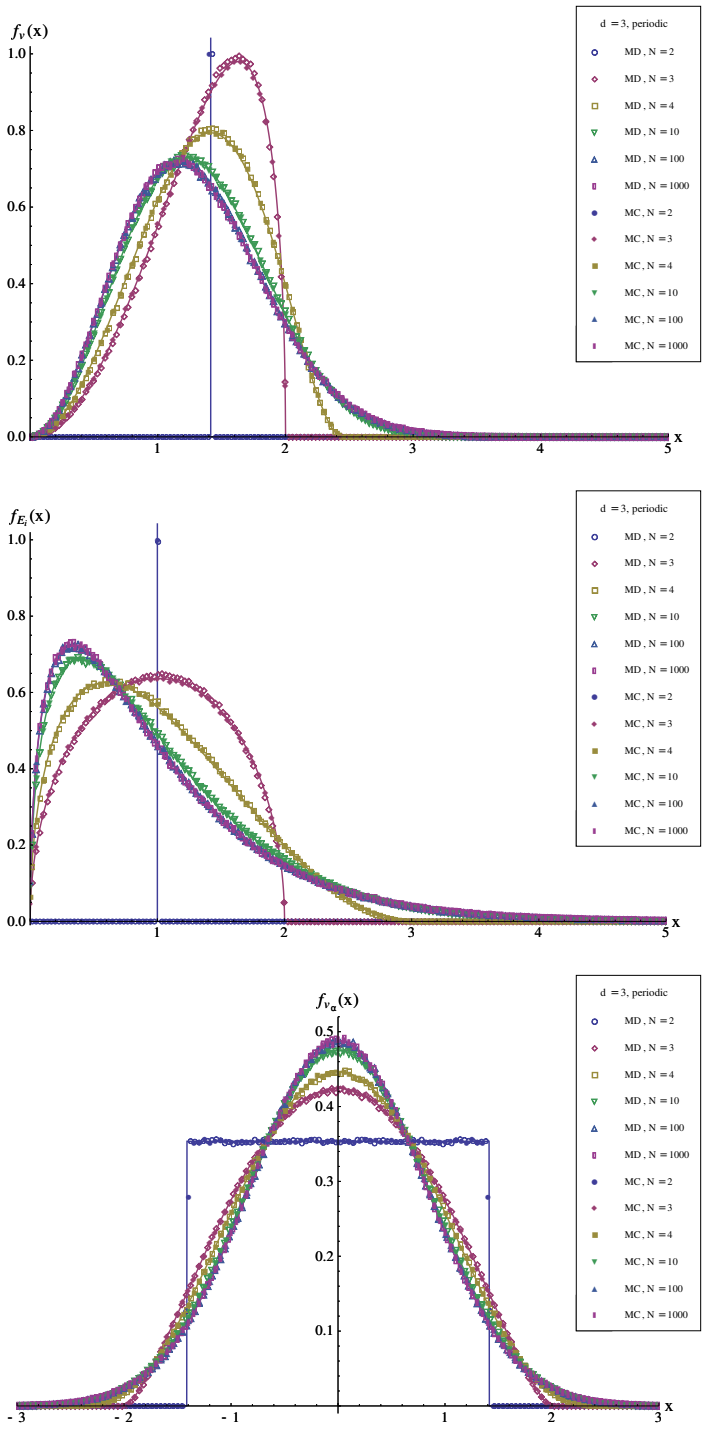


Abbildung 3.8: Diese Abbildung zeigt die Verteilungen für dreidimensionale Simulationsboxen mit periodischen Randbedingungen. Oben ist die Verteilung der Geschwindigkeitsbeträge der harten Kugeln zu sehen. Die mittlere Abbildung zeigt die Verteilung der Teilchenenergien und die untere die Verteilung der Geschwindigkeitskomponenten. Auch für dreidimensionale Simulationsboxen mit periodischen Randbedingungen kommt es zu kleinen Abweichungen in den Verteilungen der Geschwindigkeitsbeträge und Energien, während die der Geschwindigkeitskomponenten davon unbeeinflusst erscheinen. Der Effekt ist insgesamt schwächer ausgeprägt als für den Fall zweier Dimensionen, kann aber für $N = 3$ sehr deutlich den Graphen entnommen und für $N = 4$ zumindest noch erahnt werden. Darüber hinaus ist er nicht mehr zu erkennen.

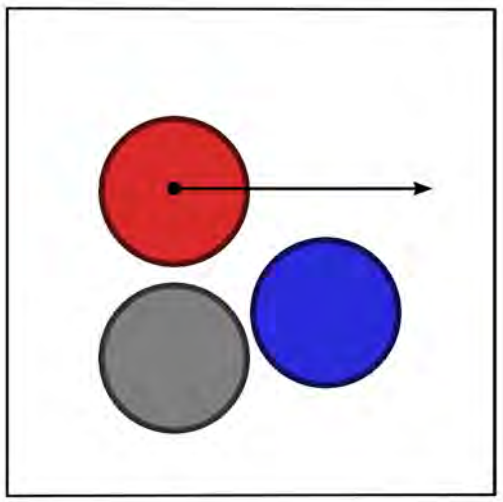


Abbildung 3.9: Diese Abbildung zeigt die Anfangskonfiguration eines Systems mit drei harten Scheiben in einer quadratischen, hartwandigen Simulationsbox mit einer Kantenlänge von etwa 3.2 Teilchendurchmessern. Dieses System soll dazu dienen, die Auswertung der Simulation und aller in deren Verlauf erstellten Graphen zu erklären. Die Wahl ist nicht zufällig erfolgt. Dieses System weist keinerlei von der Erwartung abweichendes Verhalten auf, was von den meisten anderen nicht behauptet werden kann.

Abbn. 3.11 und 3.12. Ebenfalls können die Verteilungen der Impulse der einzelnen Teilchen betrachtet werden. Diese sind in Abb. 3.13 zu sehen. Für den Gesamtimpuls muss für ein System mit HW prinzipiell keine Impulserhaltung gelten, da die Wände bei Kollisionen den Impuls aufnehmen. Insofern werden in solchen Systemen in aller Regel breit verteilte Gesamtimpulse auftreten. Gleiches kann auch über die komponentenweisen Verteilungen gesagt werden, mit der Ausnahme, dass in einem zweidimensionalen System der Impuls in $z-$Richtung natürlich zu allen Zeiten Null ist. Dieses Verhalten kann auch genau so aus den Abbn. 3.11 und 3.12 abgelesen werden. Im Falle eines Systems mit PR hingegen muss der Gesamtimpuls für alle Zeiten erhalten sein, da bei einem elastischen Teilchen-Teilchen-Stoß immer Impulserhaltung gilt, aber die Stöße mit den Wänden natürlich fehlen. Ohne letztere kann aber kein Impuls aufgenommen werden, und der Gesamtimpuls bleibt somit erhalten.

Für die Verteilung der Einzelimpulse hingegen ist im Allgemeinen immer eine gewisse Breite der Verteilung zu erwarten. Auch dieses Verhalten kann sofort aus Abb. 3.13 ersehen werden.

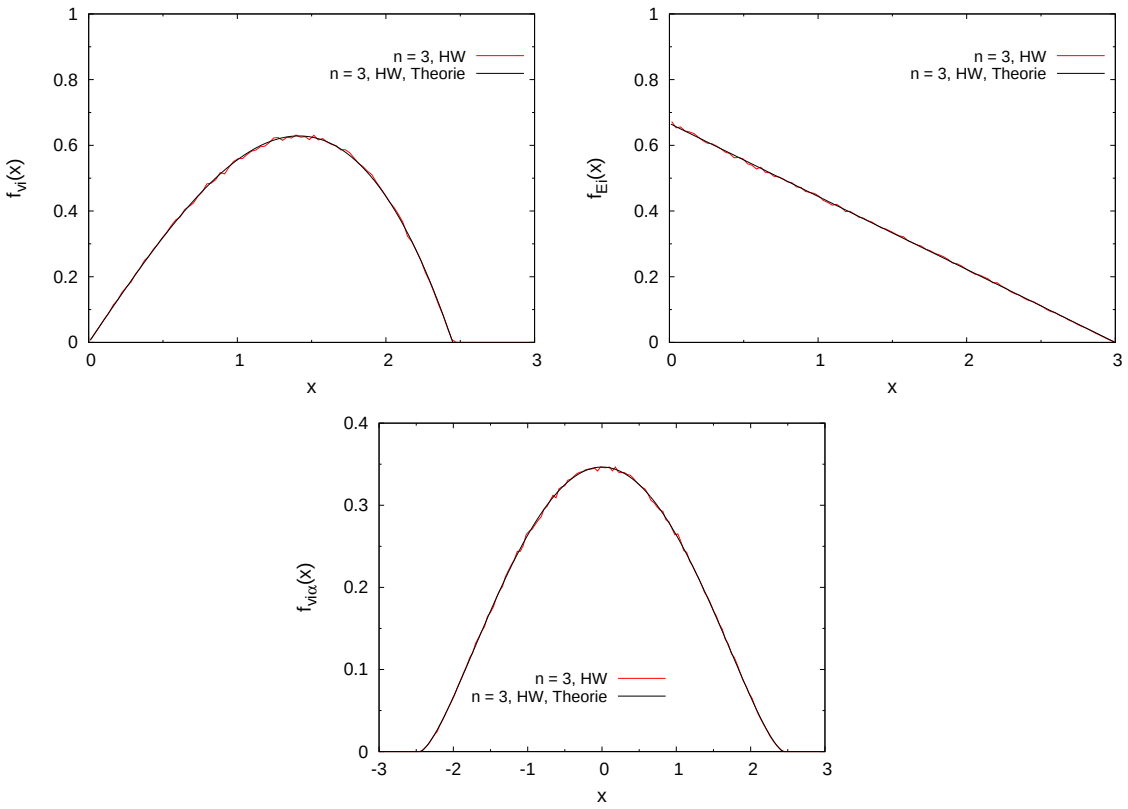


Abbildung 3.10: Diese Abbildung zeigt die Graphen, welche im Rahmen dieser Arbeit in analytischer Form reproduziert werden sollen. Oben links ist die Verteilung der Geschwindigkeitsbeträge zu sehen. Deutlich zu erkennen ist, dass diese Verteilung sicherlich nicht Maxwell-Boltzmann entspricht, wie es für den Fall vieler Teilchen zu erwarten wäre. Oben rechts ist die Verteilung der Teilchenenergien dargestellt, während in der unteren Zeile die Verteilung der einzelnen Geschwindigkeitskomponenten zu sehen sind.

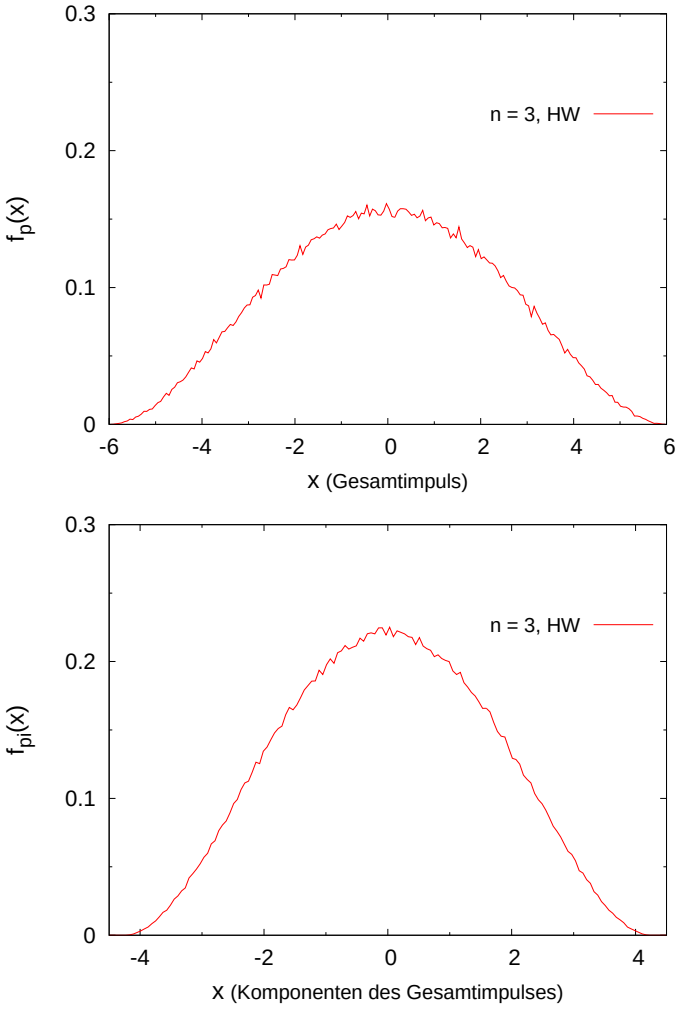


Abbildung 3.11: Diese Abbildung zeigt die Verteilungsdichten des Gesamtimpulses des Systems (oben) und die Verteilungen der Komponenten des Gesamtimpulses (unten).

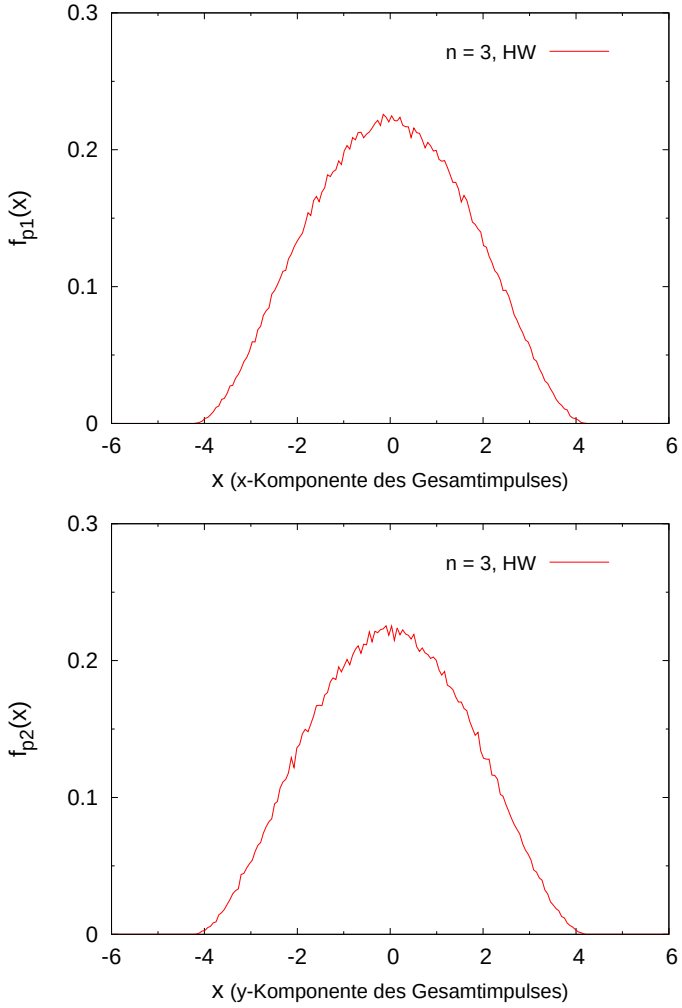


Abbildung 3.12: Die Abbildung zeigt die Verteilungsdichten der Komponenten des Gesamtimpulses des Systems: Oben die Verteilung der $x-$Komponenten und unten die der $y-$Komponenten des Gesamtimpulses. Da es sich um ein System mit harten Wänden handelt, die in der Lage sind Impuls aufzunehmen, kann davon ausgegangen werden, dass bei einer ungeraden Anzahl von Scheiben der Gesamtimpuls nicht erhalten bleibt.

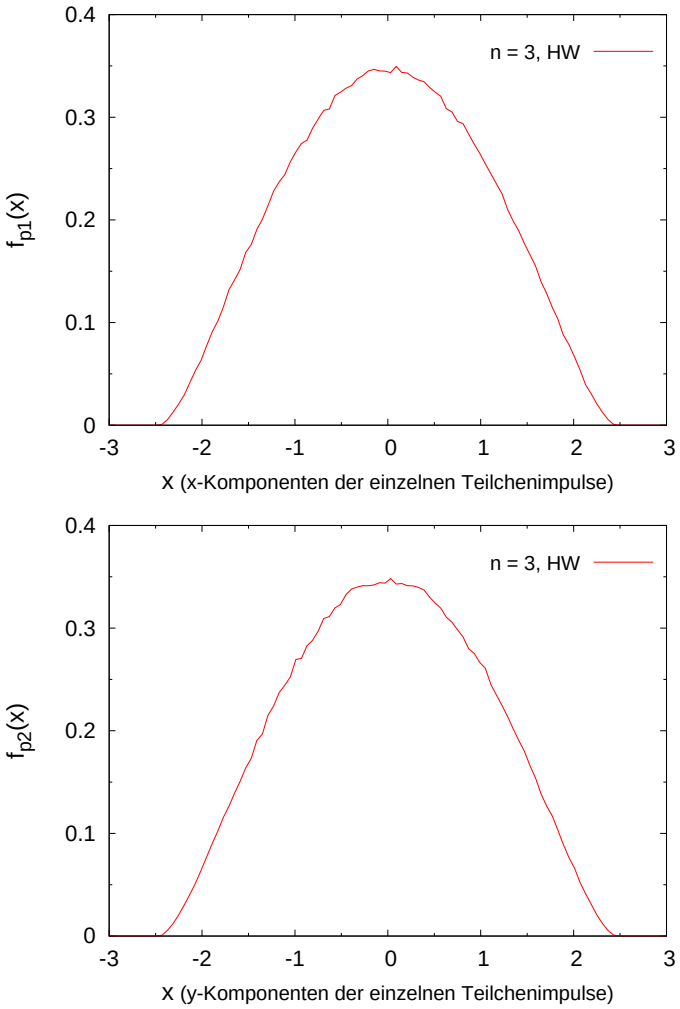


Abbildung 3.13: Diese Abbildung zeigt die Verteilungsdichten der Komponenten der Teilchenimpulse: Oben ist die Verteilungsdichte der $x-$Komponenten und unten die der $y-$Komponenten der Teilchenimpulse dargestellt. Eine gewisse Verteilung in den komponentenweisen Teilchenimpulsen sollte immer zu erwarten sein, selbst wenn insgesamt Impulserhaltung gefordert werden kann, wie zum Beispiel bei Systemen mit periodischen Randbedingungen.

3.6.1 Auffälliges Verhalten bei drei harten Scheiben in Box mit PR

Die Verteilungen der Energien und Geschwindigkeitsbeträge zeigen bei Systemen mit drei harten Scheiben in einer Simulationsbox mit PR ein leicht von der Theorie abweichendes Verhalten. Zu ihrer Untersuchung wurden vier Systeme herangezogen, welche in Abb. 3.14 dargestellt sind. Dabei ist zu beachten, dass das System oben links zweimal simuliert wurde, das erste Mal bis eine Millionen Kollisionen erreicht waren, und ein weiteres Mal bis zehn Millionen Kollisionen. Die Abweichung selbst kann in Abb. 3.15 betrachtet werden. Für die Verteilungen der Geschwindigkeitsbeträge wird ein scharfes Dreieck erwartet und für die Verteilungen der Energien ein Rechteck. Es fällt auf, dass sowohl das Dreieck als auch das Rechteck grundsätzlich vorhanden sind, beide aber leicht von der idealen Form abweichen. Weiterhin kann innerhalb der an dieser Stelle getätigten Variationen keinerlei voneinander abweichendes Verhalten festgestellt werden. Es ist somit notwendig, weitere Berechnungen anderer Systeme, auch mit unterschiedlicher Teilchenzahl, zu betrachten und auf das Vorhandensein dieser oder ähnlicher Abweichungen hin zu untersuchen.

Da es sich bei allen Systemen um solche mit PR handelt, sollte der Gesamtimpuls zu Null erhalten bleiben. Die einzelnen Impulse der Teilchen hingegen sollten eine breite Verteilung aufweisen, welche sich allerdings in $x-$ und $y-$Richtung nicht unterscheiden sollte. Diese Gegebenheiten können zur Konsistenzprüfung den Abbn. 3.16 und 3.17 entnommen werden und sind erfüllt.

3.6.2 Vergleich des Verhaltens von drei harten Scheiben mit PR und vier harten Scheiben in punktsymmetrischer Anordnung mit HW

Die grundsätzliche Form der betrachteten Verteilungen hängt lediglich von der Anzahl der Teilchen und der Dimensionalität des System ab. Innerhalb einer Dimensionalität hängt die Form der Verteilungen ausschließlich von der Anzahl der Freiheitsgrade im System ab. Dieses Verhalten soll in den nächsten Abschnitten behandelt werden. Die einzigen Freiheitsgrade, die ein Teilchen in unserem Zusammenhang beisteuert, rühren von der Energie der Bewegung her, da weder ein äußeres noch ein inneres Potential wirkt; Rotationen sind ebenfalls ausgeschlossen. Ein einzelnes Teilchen besitzt also eine Anzahl an Freiheitsgraden, die der Anzahl der Raumdimensionen des Systems entspricht. Diese Aussage ist allerdings nur dann gültig, wenn die Teilchen keinen Zwangsbedingungen unterworfen sind. Im vorliegenden Fall der vier harten Scheiben in einer Box mit HW werden durch die punktsymmetrische Anordnung der Startpositionen und Startgeschwindigkeiten Zwangsbedingungen in das System eingebracht. Bleibt die Punktsymmetrie erhalten, bewegen sich zu allen Zeiten die Scheiben paarweise symmetrisch zueinander, und die effektive

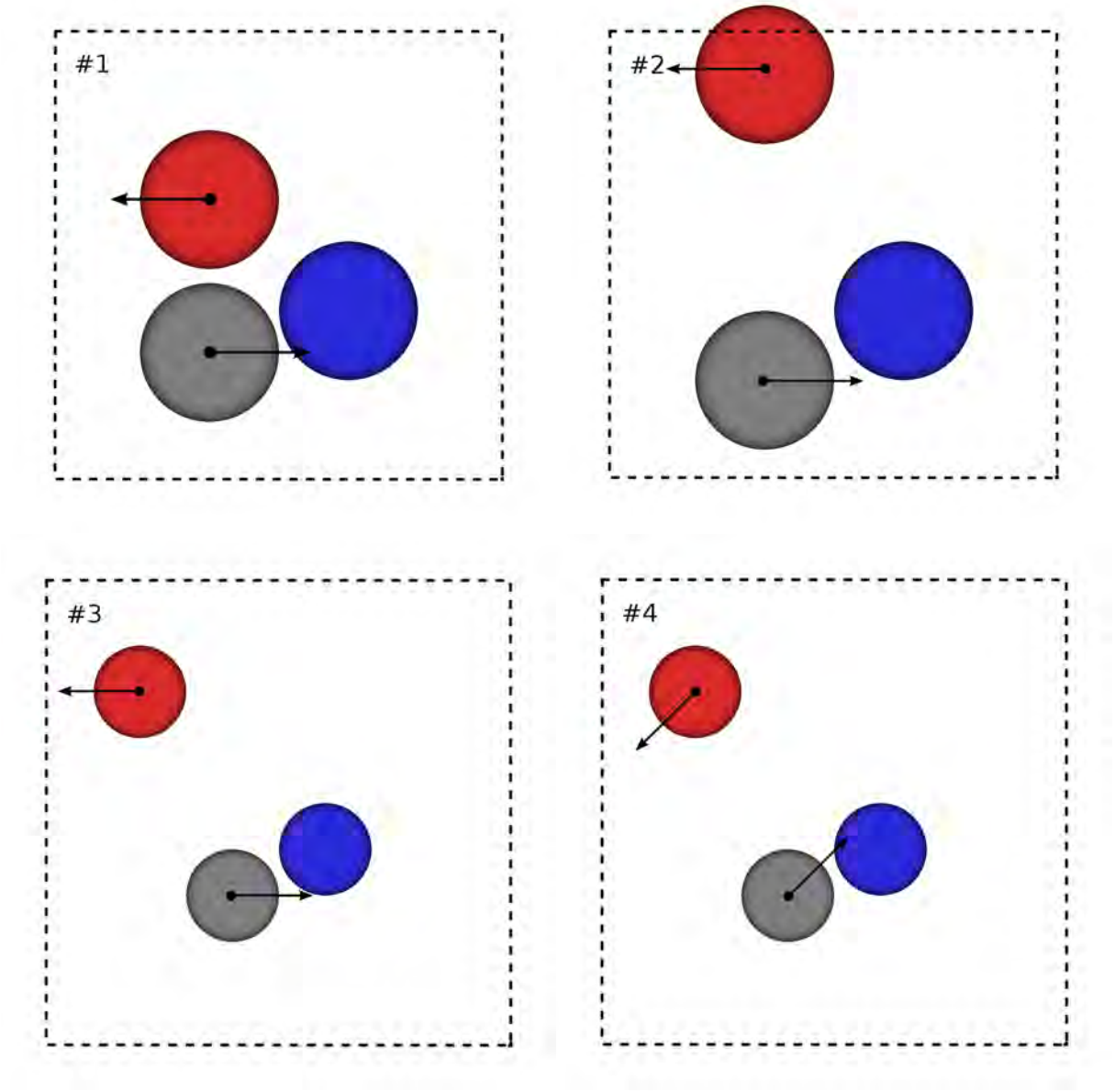


Abbildung 3.14: Diese Abbildung zeigt die vier Systeme, welche zur Untersuchung des abweichenden Verhaltens dreier harter Scheiben in einer Simulationsbox mit periodischen Randbedingungen berechnet wurden. Im in den folgenden Abbildungen der Verteilungen werden die Systeme folgendermaßen bezeichnet: Oben links mit #1, oben rechts mit #2, unten links mit #3 und unten rechts mit #4. Außerdem taucht noch ein System #5 auf. Bei diesem handelt es sich um #1, allerdings mit einer um den Faktor zehn größeren Simulationsdauer (zehn Millionen Kollisionen). Variiert wurden bei der Auswahl der Systeme die Startposition von #1 nach #2, die Boxgröße und die Startposition von #2 nach #3 und die Anfangsgeschwindigkeiten von #3 nach #4. Alle Systeme starten mit einem Gesamtimpuls von Null.

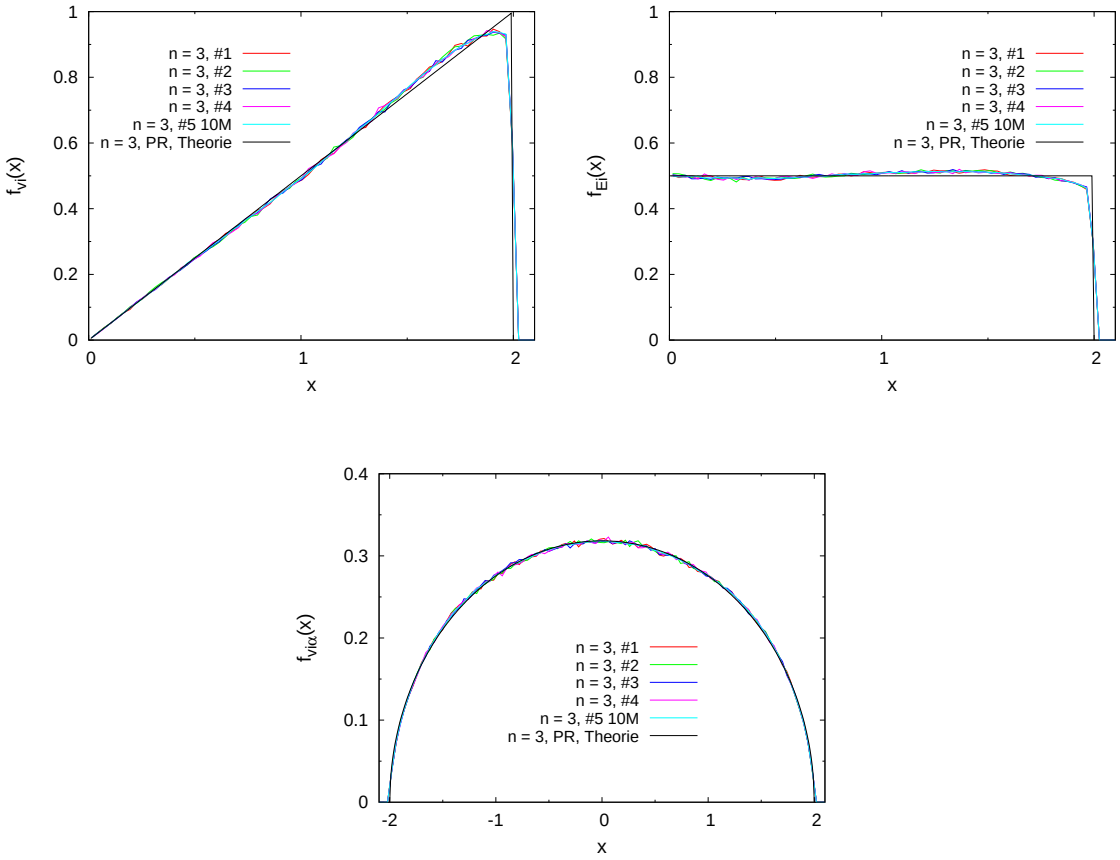


Abbildung 3.15: Verteilungen der Geschwindigkeitsbeträge (oben links), der Energien (oben rechts) und der Geschwindigkeitskomponenten (unten) von harten Scheiben bei periodischen Randbedingungen. Während die Verteilungen für die Geschwindigkeitskomponenten der theoretischen Erwartung entsprechen weichen die beiden anderen leicht von dieser ab.

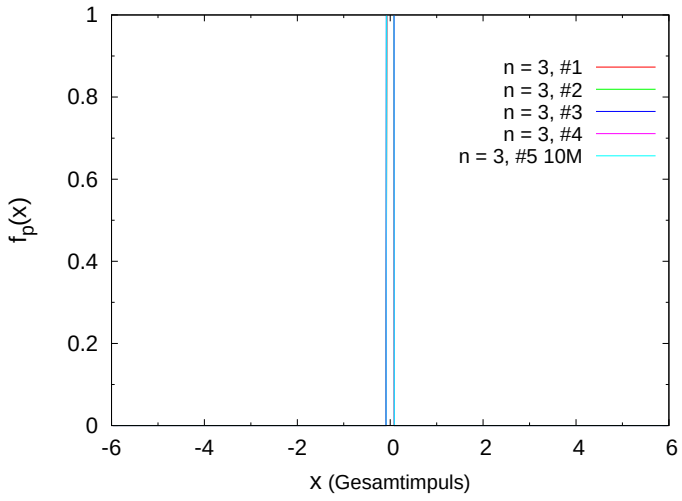


Abbildung 3.16: Bei allen fünf Systemen handelt es sich um solche mit periodischen Randbedingungen. Damit konsistent ist die Aussage dieser Abbildung, dass der Gesamtimpuls zu allen Zeiten den Wert null annimmt.

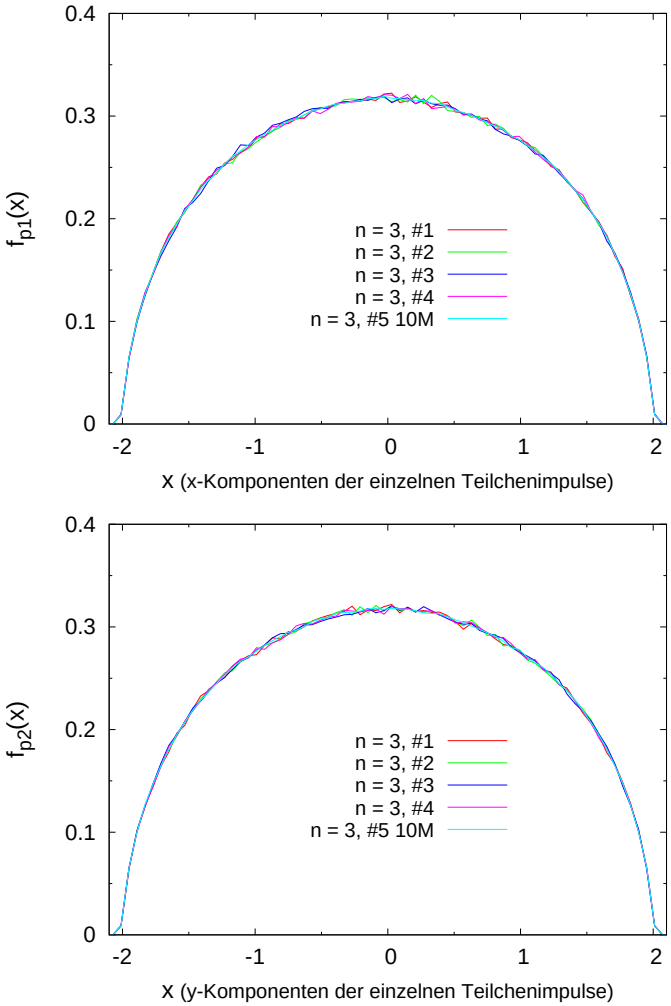


Abbildung 3.17: Diese Abbildung zeigt die Verteilung der Teilchenimpulse getrennt in $x-$ und $y-$Komponenten. Die Verteilungen sind wie erwartet nicht z.B. $\delta-$förmig, sondern verbreitert und für beide Richtungen von gleicher Gestalt.

Anzahl unabhängiger Teilchen g wäre nur halb so groß wie die tatsächliche, das heißt $g = N/2$. Die vier Scheiben dieses zweidimensionalen Systems sollten sich also verhalten wie zwei unabhängige Scheiben. Ebenso muss für ein solches System Impulserhaltung gelten, da ansonsten die Symmetrie nicht erhalten bleiben könnte.

Eine ähnliche Argumentation kann für das periodische System mit drei Scheiben geführt werden. In PR muss der Gesamtimpuls zu allen Zeiten erhalten bleiben, da es keine Wände gibt, die den Impuls aufnehmen können, und der Paarprozess des Teilchenstoßes per se impulserhaltend ist. Würden nun zwei Teilchen mit beliebigen Positionen und Impulsen in die Box gegeben, dann müsste der Impuls des dritten Teilchens den Gesamtimpuls der ersten beiden kompensieren. Dies ergibt aber eine Zwangsbedingung, welche zu jeder Zeit die effektive Teilchenzahl um eins erniedrigt. Das periodische System mit drei Scheiben sollte sich demnach ebenfalls verhalten wie ein System mit zwei unabhängigen Scheiben. Die Argumentation kann leicht auf beliebige Teilchenzahlen N erweitert werden und führt für PR zu einer effektiven Zahl unabhängiger Teilchen von $g = N - 1$. Die Startkonfigurationen der Systeme sind in Abb. 3.18 dargestellt. Die Impulserhaltung trotz HW kann ebenfalls bestätigt werden, siehe dazu Abb. 3.20.

Die Auffälligkeit in den Verteilungen, wie sie im vorangegangenen Abschnitt untersucht wurde, kann auch hier, wie erwartet, für das Dreiersystem beobachtet werden. Interessant ist jedoch, dass diese Auffälligkeit ebenfalls für das symmetrische Vierersystem mit HW auftritt, wenn auch die Form der Abweichung sich unterscheidet. Siehe dazu Abb. 3.19 zum Vergleich. Es ist auch für das Vierersystem zu beobachten, dass die Abweichung lediglich die Energieverteilung und die Verteilung der Geschwindigkeitsbeträge betrifft, während die der Geschwindigkeitskomponenten den Erwartungen entspricht.

3.6.3 Vergleich des Verhaltens von vier harten Scheiben mit PR und sechs harten Scheiben in punktsymmetrischer Anordnung mit HW

Sind die Erkenntnisse des letzten Abschnitts korrekt, so müssen sich ein System mit PR und vier harten Scheiben und ein System mit HW und sechs harten Scheiben in punktsymmetrischer Startkonfiguration gleich verhalten. Für das erste System beträgt die Anzahl unabhängiger Teilchen $g_1 = 4 - 1 = 3$. Für das zweite gilt $g_2 = \frac{6}{2} = 3$. Da somit die Anzahl der Freiheitsgrade beider Systeme identisch ist, kann davon ausgegangen werden, das sie sich im Verhalten gleichen werden. In Abb. 3.21 sind die Startkonfigurationen beider Systeme graphisch dargestellt und Abb. 3.22 bestätigt eindeutig die Annahme gleichen Verhaltens im Sinne der Verteilungen für Energie, Geschwindigkeitsbeträge und Geschwindigkeitskomponenten der Teilchen. Aus Abb. 3.23 kann die notwendige Impulserhaltung für beide Systeme abgelesen werden.

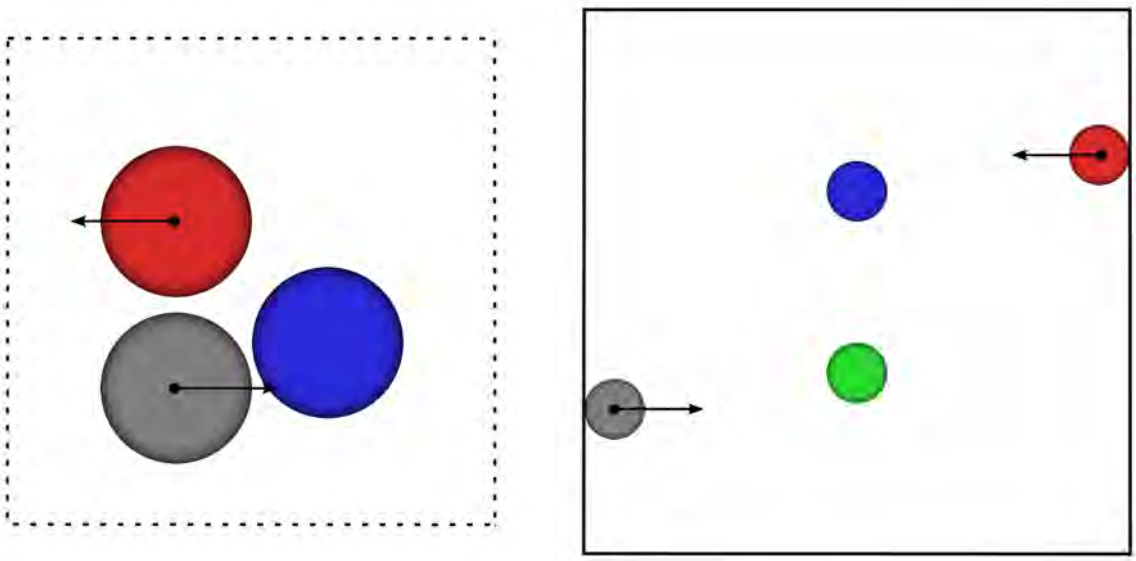


Abbildung 3.18: Diese Abbildung zeigt zwei Systeme, welche prinzipiell gleiches Verhalten zeigen, zumindest was die generelle Form der Verteilung der Energie, der Geschwindigkeitsbeträge und der Geschwindigkeitskomponenten angeht. Der Impuls bleibt in beiden Systemen erhalten, was für das linke gefordert, für das rechte aber eher unerwartet ist. Bei dem linken System handelt es sich um drei harte Scheiben in einer Box mit periodischen Randbedingungen. Das rechte System besteht aus vier harten Scheiben in einer Box mit harten Wänden, wobei die Anfangspositionen und die Anfangsgeschwindigkeiten absolut punktsymmetrisch zum Mittelpunkt der Simulationsbox gewählt sind. Allein die Punktsymmetrie sorgt für das gleiche Verhalten dieser beiden unterschiedlichen Systeme.

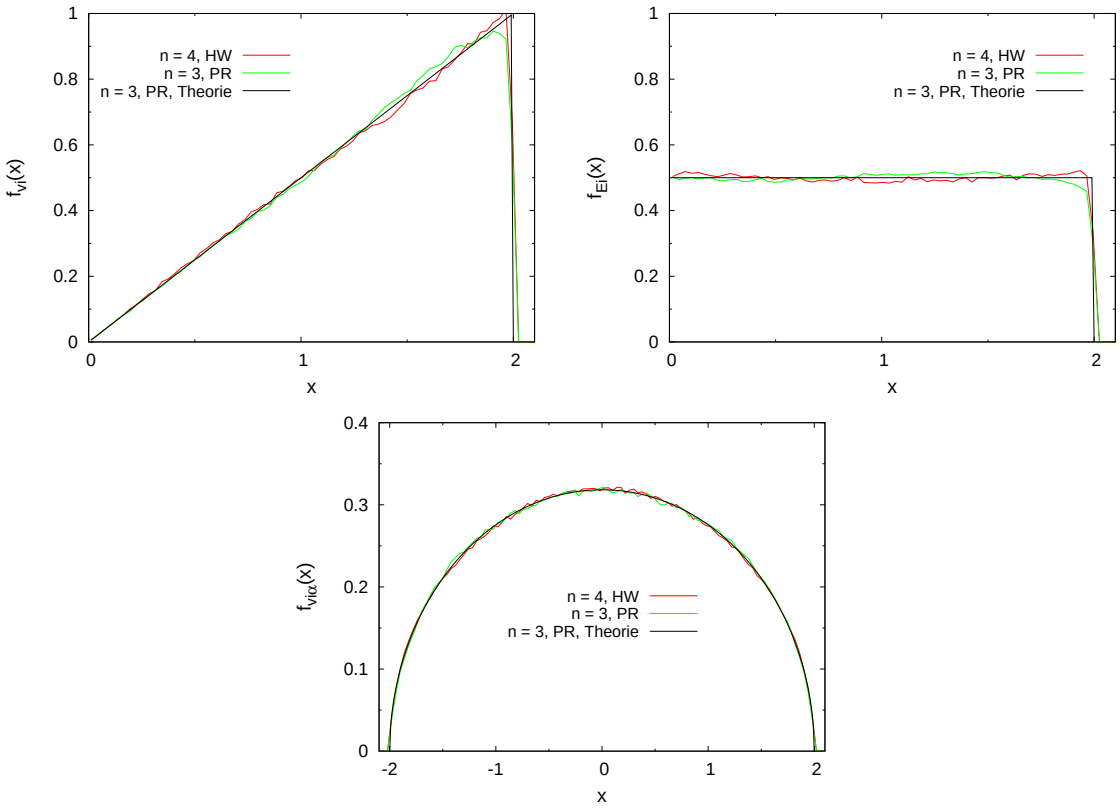


Abbildung 3.19: Diese Abbildung zeigt die Verteilungen der Geschwindigkeitsbeträge (oben links), der Energien (oben rechts) und der Geschwindigkeitskomponenten (unten) der harten Scheiben. Zu erkennen ist zum einen, dass die Verteilungen prinzipiell von gleicher Form sind, obwohl sich die Systeme voneinander unterscheiden. Für die Übereinstimmung ist die Wahl der Anfangskonfiguration des Vierersystems verantwortlich. Ebenfalls auffällig ist die Tatsache, dass die schon im vorangegangenen Abschnitt thematisierte Abweichung für die Verteilung der Energie und Geschwindigkeitsbeträge in beiden Systemen auftritt, auch wenn sie von anderer expliziter Gestalt ist.

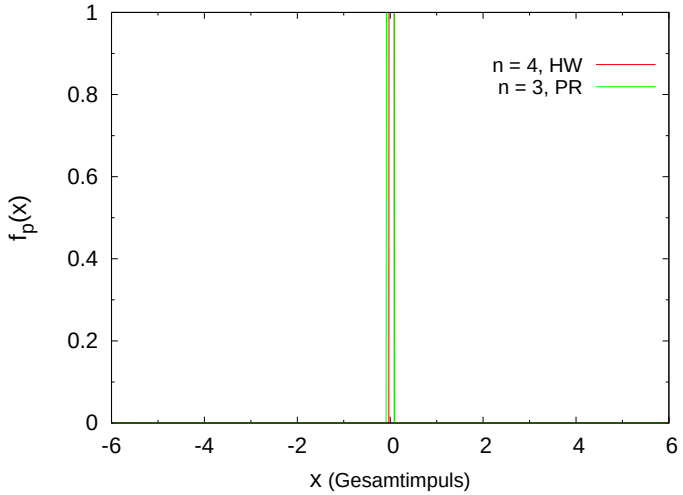


Abbildung 3.20: Diese Abbildung zeigt die Verteilung der Gesamtimpulse beider Systeme. Es ist deutlich zu erkennen, dass der Gesamtimpuls in beiden Systemen erhalten bleibt.

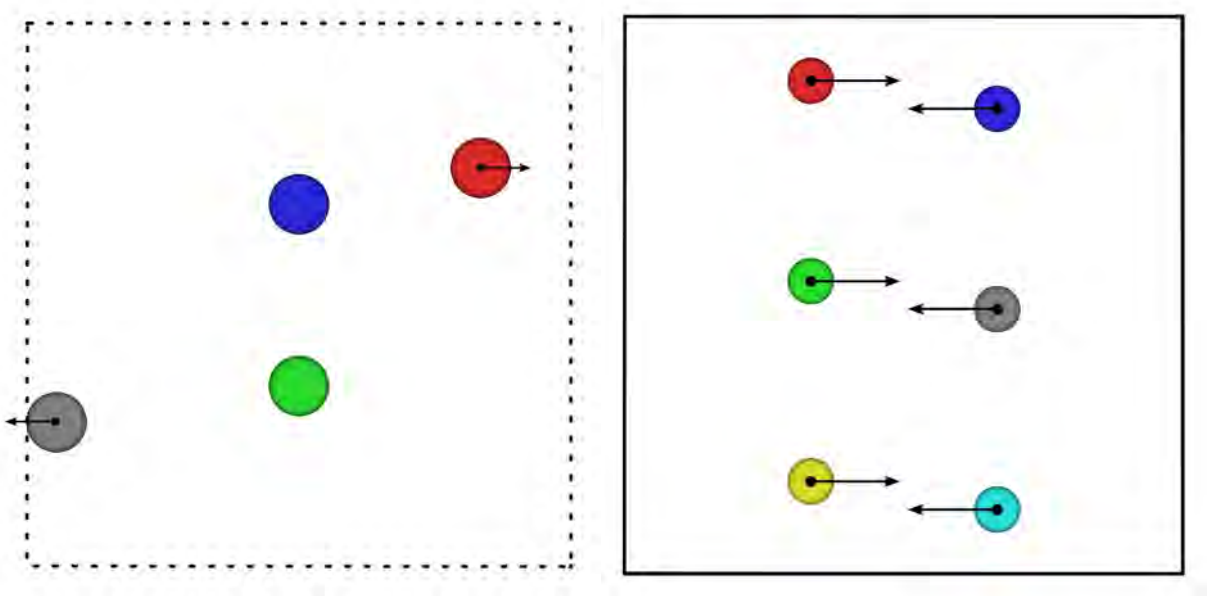


Abbildung 3.21: Diese Abbildung zeigt die Anfangskonfigurationen der in Abschnitt 3.6.3 behandelten Systeme. Links sind vier harte Scheiben in einer Box mit periodischen Randbedingungen zu sehen und rechts sechs harte Scheiben in einer Box mit harten Wänden. Im rechten System sind zusätzlich die Startkonfiguration punktsymmetrisch zum Mittelpunkt der Box gewählt. Daraus resultiert gleiches Verhalten der Verteilungen der Energie, der Geschwindigkeitsbeträge und der Geschwindigkeitskomponenten. Dies ist, wie schon im vorigen Abschnitt gezeigt, mit der gleichen Anzahl unabhängiger Teilchen und damit Freiheitsgraden in den Systemen begründet.

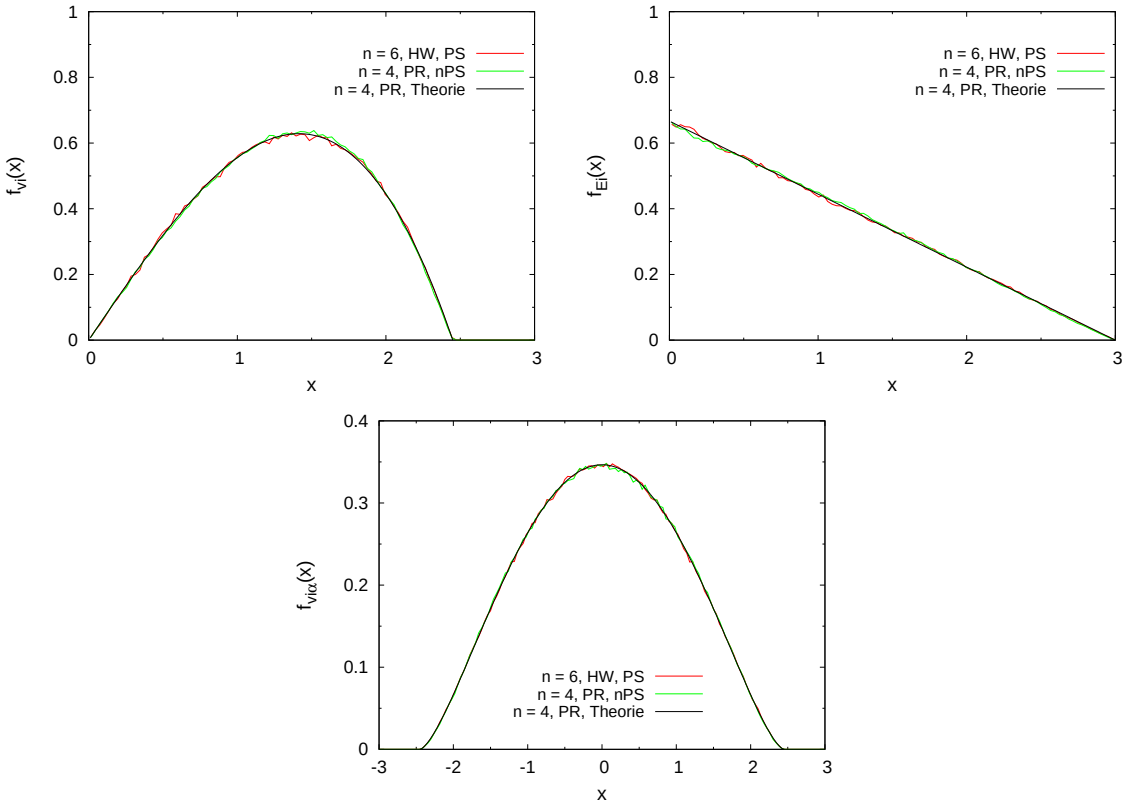


Abbildung 3.22: Diese Abbildung zeigt die Verteilungen der Geschwindigkeitsbeträge (oben links), der Energien (oben rechts) und der Geschwindigkeitskomponenten (unten) von harten Scheiben aus Abschnitt 3.6.3. Zu erkennen ist zum einen, dass die Verteilungen prinzipiell von gleicher Form sind, obwohl die Systeme sich voneinander unterscheiden. Für die Übereinstimmung ist die Symmetrie der Anfangskonfiguration des Sechsersystems verantwortlich.

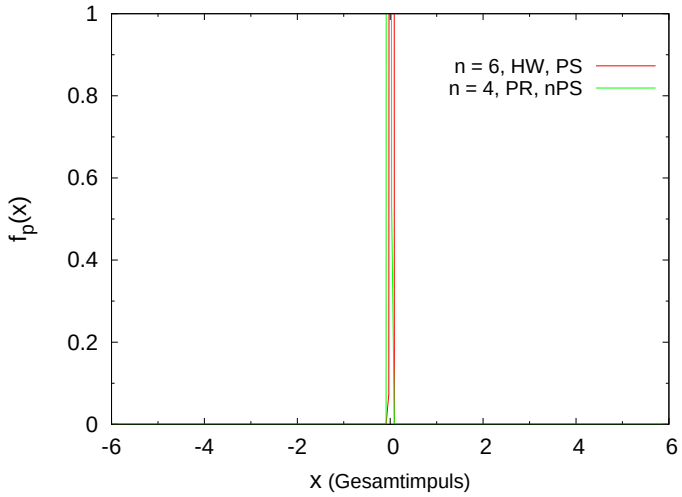


Abbildung 3.23: Diese Abbildung zeigt die Verteilung der Gesamtimpulse beider Systeme aus Abschnitt 3.6.3. Es ist deutlich zu erkennen, dass der Gesamtimpuls in beiden Systemen erhalten bleibt.

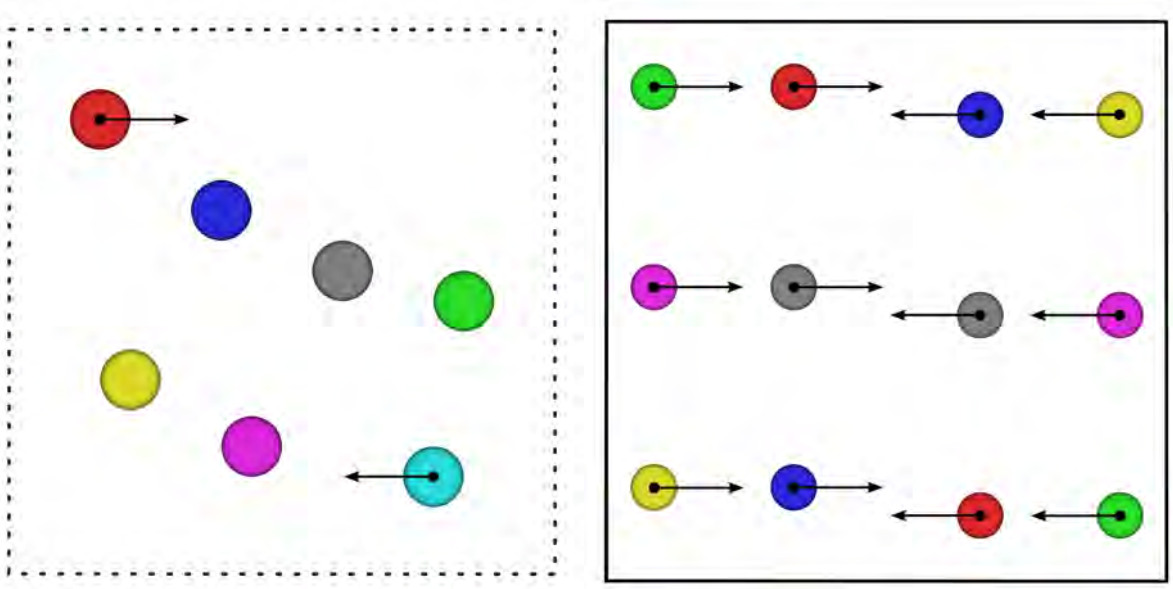


Abbildung 3.24: Diese Abbildung zeigt die Anfangskonfigurationen der in Abschnitt 3.6.4 behandelten Systeme. Links sind sieben harte Scheiben in einer Box mit periodischen Randbedingungen zu sehen und rechts zwölf harte Scheiben in einer Box mit harten Wänden. Im rechten System sind zusätzlich die Startkonfiguration punktsymmetrisch zum Mittelpunkt der Box gewählt. Daraus resultiert gleiches Verhalten der Verteilungen für die Energie, die Geschwindigkeitsbeträge und die Geschwindigkeitskomponenten. Dies ist, wie schon im vorigen Abschnitt gezeigt, mit der gleichen Anzahl unabhängiger Teilchen und damit Freiheitsgraden in den Systemen begründet.

3.6.4 Vergleich des Verhaltens von sieben harten Scheiben mit PR und zwölf harten Scheiben in punktsymmetrischer Anordnung mit HW

Als Abschluss zu den vorangegangenen Abschnitten sollen nun ein letztes mal zwei Systeme verglichen werden, die sich in ihrem Verhalten gleichen sollten. Dabei handelt es sich zum einen um sieben harte Scheiben in einer Box mit PR und dementsprechend $g = 7 - 1 = 6$ und zum anderen um eine punktsymmetrische Anordnung von zwölf harten Scheiben in einer Box mit HW und ebenfalls $g = 12/2 = 6$. Die Startkonfigurationen sind in Abb. 3.24 graphisch dargestellt. Die betrachteten Verteilungen zeigt Abb. 3.25 und bestätigt eindeutig die Vermutung gleichen Verhaltens der beiden Systeme. Die zur Kontrolle herangezogene Verteilung des Gesamtimpulses zeigt für beide Systeme Impulserhaltung und ist in Abb. 3.26 dargestellt.

3.6.5 Das Verhalten zweier harter Scheiben in einer Box mit HW

In diesem Abschnitt wird auf das System eingegangen, welches von den Untersuchten das auffälligste Verhalten zeigt. Überraschenderweise handelt es sich dabei auch um das am einfachsten strukturierte System, bestehend aus zwei harten Scheiben in einer hart-

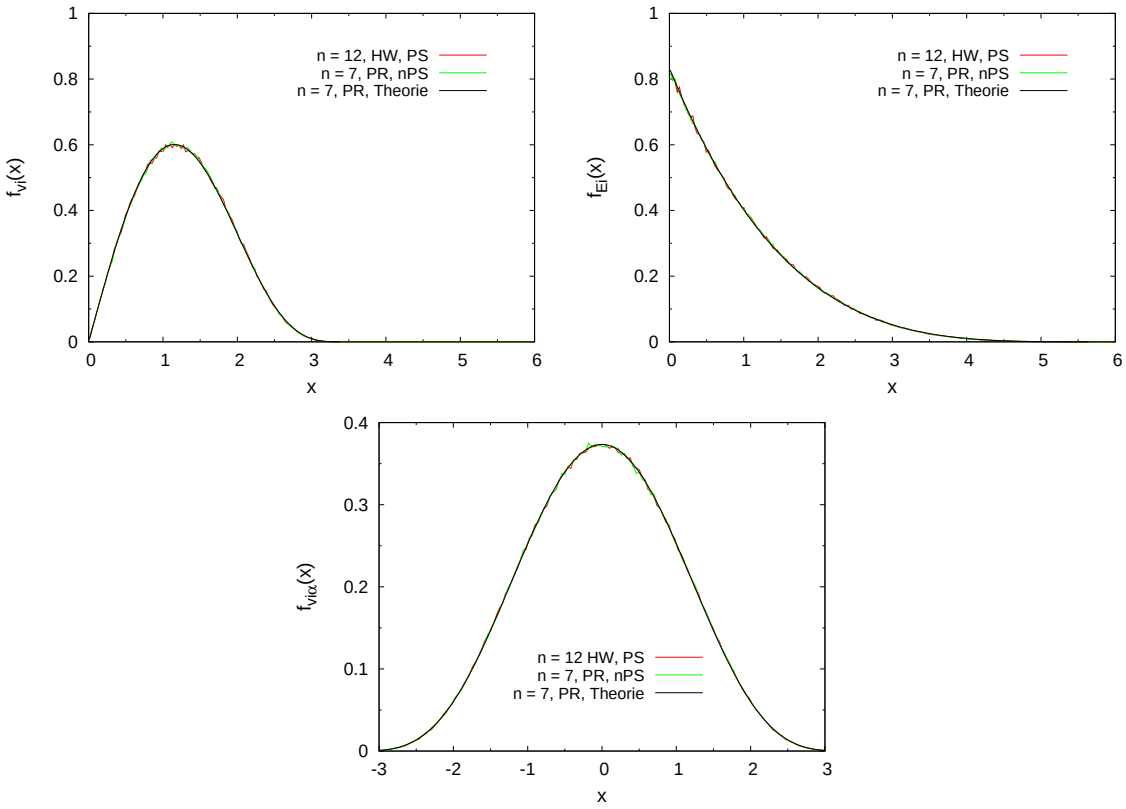


Abbildung 3.25: Diese Abbildung zeigt die Verteilungen der Geschwindigkeitsbeträge (oben links), der Energien (oben rechts) und der Geschwindigkeitskomponenten (unten) der harten Scheiben. Zu erkennen ist zum einen, dass die Verteilungen prinzipiell von gleicher Form sind, obwohl die Systeme sich voneinander unterscheiden. Für die Übereinstimmung ist die Symmetrie der Anfangskonfiguration des Zwölfersystems verantwortlich.

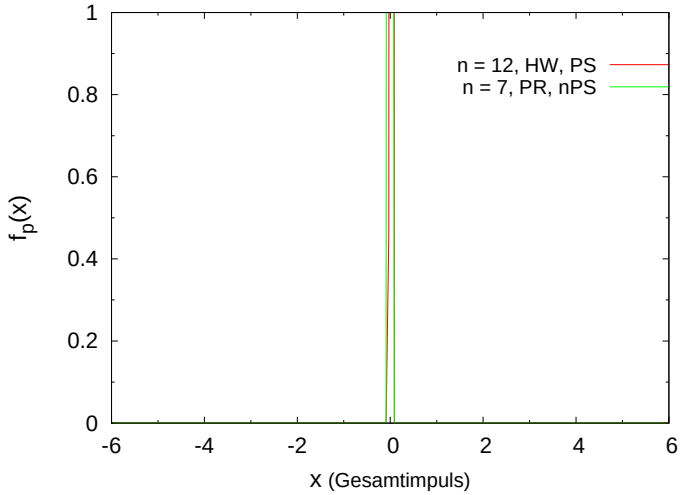


Abbildung 3.26: Diese Abbildung zeigt die Verteilung der Gesamtimpulse beider Systeme. Es ist deutlich zu erkennen, dass der Gesamtimpuls in beiden Systemen erhalten bleibt.

wandigen Simulationsbox. Die verschiedenen untersuchten Startkonfigurationen sind in Abb. 3.27 zu sehen. Von den dort abgebildeten Konfigurationen starten zwei mit einem Gesamtimpuls, der parallel zu einer Boxkante ausgerichtet ist. Daraus folgt, dass eine der Komponenten des Gesamtimpulses zu Beginn den Wert Null annimmt, da die Kanten der Simulationsbox parallel bzw. senkrecht zu den Koordinatenachsen ausgerichtet sind. Es zeigt sich im Laufe der Simulation, dass dieser Wert erhalten bleibt, das heißt, die Komponenten des Gesamtimpulses weisen eine unterschiedliche Verteilung in den verschiedenen Raumrichtungen auf.

Der Gesamtimpuls der beiden übrigen Systeme schließt zu Simulationsbeginn einen Winkel von einmal 45° und einmal 1° mit der horizontalen Koordinatenachse ein. Unter diesen Bedingungen verschwindet die partielle Impulserhaltung und die berechneten Verteilungen entsprechen in etwa den Erwartungen, jedoch kommt es auch an dieser Stelle zu Abweichungen in deren expliziter Form, siehe dazu Abb. 3.28. Es fällt auf, dass in diesem Fall auch die Verteilungen der Geschwindigkeitskomponenten betroffen ist. Bei der eingangs beobachteten Abweichung für drei Scheiben in PR schien dies nicht der Fall zu sein. An dieser Stelle ist die Abweichung hingegen augenfällig und wurde auch durch eine um den Faktor zehn erhöhte Simulationsdauer nicht widerlegt.

Zur weiteren Analyse des Verhaltens dieser Systeme werden nach Raumrichtung getrennte Verteilungen der Gesamt- und Teilchenimpulse herangezogen. Diese sind in den Abb. 3.29 und 3.30 dargestellt. Eine eingehende Betrachtung dieser Verteilungen kann die Eigenarten dieser Systeme aufklären. In den räumlich getrennten Gesamtimpulsverteilungen tritt eine partielle Impulserhaltung der jeweils verschwindenden Komponente für die beiden ersten Systeme zu Tage. Daraus resultieren ebenfalls unterschiedliche Verteilungen für die Raumrichtungen der Verteilungen der Teilchenimpulse. Da im Rahmen unserer Simulationen die Teilchenmasse zu $m = 1$ gesetzt wurde, gleichen sich Impuls und Geschwindigkeit. Demnach kann die Verteilung der Teilchenimpulskomponenten aus Abb. 3.28 unten verstanden werden als Summe der jeweiligen Verteilungen aus Abb. 3.30. Auf diese Weise ist auch das Zustandekommen dieser etwas überraschenden Verteilungen geklärt.

3.6.6 Bemerkungen zum dreidimensionalen Fall

Die in den vorangegangenen Abschnitten gemachten Beobachtungen treffen in ähnlicher Art und Weise auch für dreidimensionale Systeme zu. An dieser Stelle kann jedoch auf eine detaillierte Darstellung der Verhältnisse verzichtet werden, da aus ihnen kein weiterer Verständnisgewinn entspringt. Für die Endauswertung sei lediglich angemerkt, dass selbstverständlich alle dort gezeigten Ergebnisse auf realen Simulationsläufen beruhen. Dies ist insbesondere auch für den dreidimensionalen Fall zutreffend, auch wenn hier auf die ausführliche Darstellung der Startkonfigurationen verzichtet wurde.

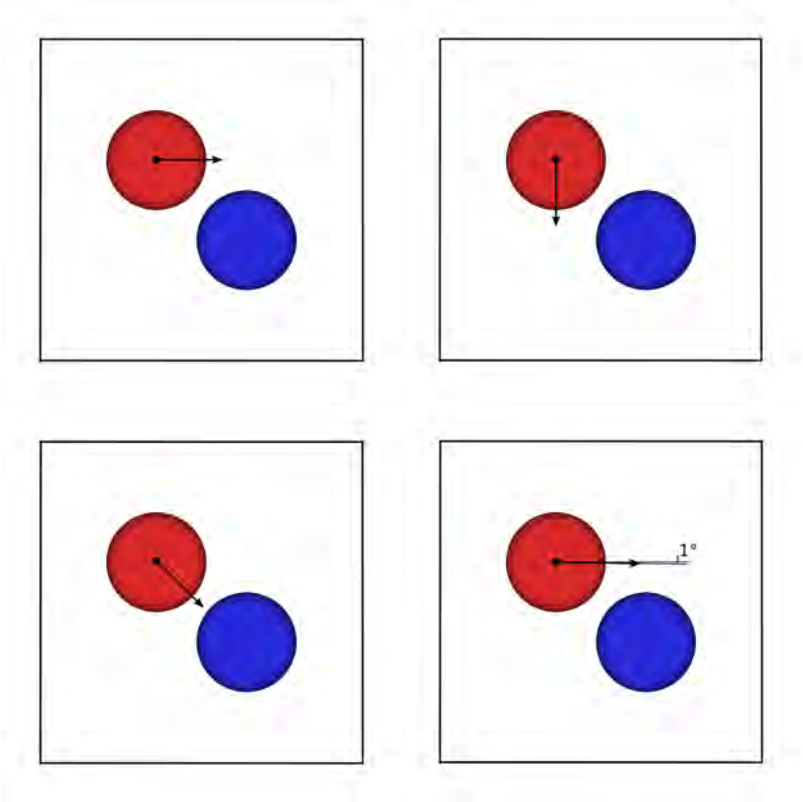


Abbildung 3.27: Diese Abbildung zeigt die Anfangskonfigurationen der in Abschnitt 3.6.5 behandelten Systeme. Bei den oberen Systemen wurde der Gesamtimpuls am Anfang parallel zu einer Koordinatenachse gewählt. Damit ergibt sich oben links der Gesamtimpuls parallel zur horizontalen Koordinatenachse und oben rechts parallel zur vertikalen. Im Laufe der Simulation zeigt sich eine daraus resultierende partielle Impulserhaltung der anfänglich verschwindenden Komponente. Der Gesamtimpuls der beiden unteren Systeme weist zu Simulationsbeginn einen Winkel von 45° (unten links) bzw. 1° (unten rechts) mit der horizontalen Koordinatenachse auf. In diesen Anordnungen verhalten sich die Systeme nahezu wie erwartet. Nahezu soll bedeuten, dass die generelle Form der Verteilungen stimmt, es aber zu Abweichungen von der idealen theoretischen Kurve kommt, siehe dazu Abb. 3.28.

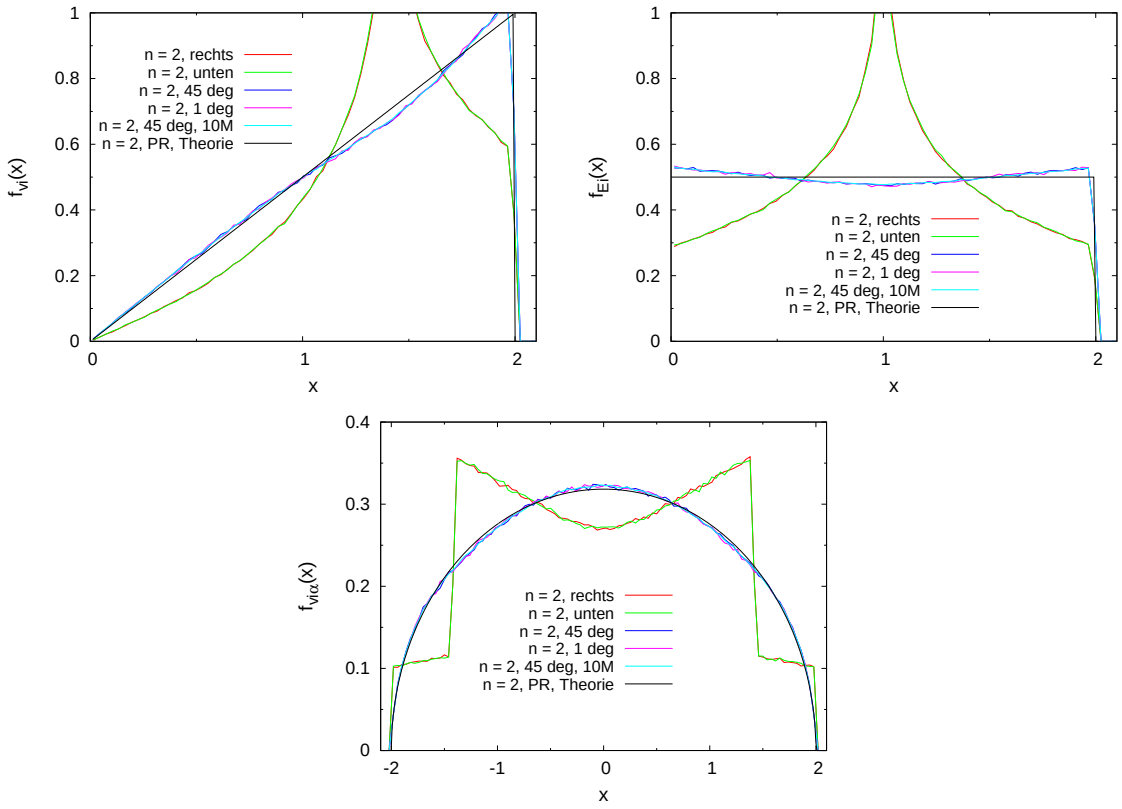


Abbildung 3.28: Diese Abbildung zeigt die Verteilungen der Geschwindigkeitsbeträge (oben links), der Energien (oben rechts) und der Geschwindigkeitskomponenten (unten) von harten Scheiben aus Abschnitt 3.6.5. Zunächst besonders auffällig ist die starke Abweichung der Verteilungen für die verschiedenen Systeme, obwohl alle prinzipiell gleicher Natur sind. Der Grund für diesen Bruch findet sich in der partiellen Impulserhaltung wie er für die beiden Systeme auftritt, deren Gesamtimpuls zu Beginn parallel zu einer Kante der Simulationsbox ausgerichtet ist. Wie es sich zeigt, bleibt die verschwindende Impulskomponente erhalten. Dies führt auch zu unterschiedlichen Verteilungen in den Raumrichtungen für die Teilchenimpulse. Eine nach Raumrichtung aufgeschlüsselte Darstellung der Gesamt- und Teilchenimpulse zeigen die Abbn. 3.29 und 3.30.

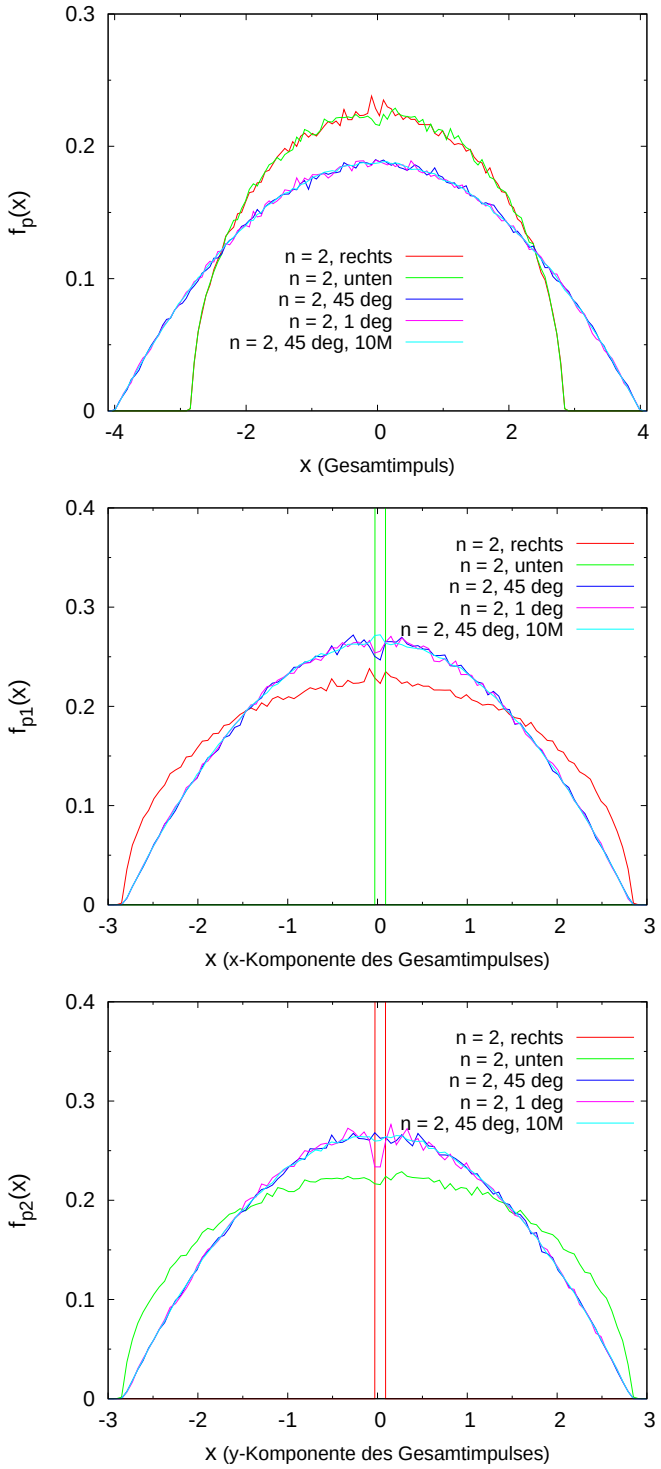


Abbildung 3.29: Diese Abbildung zeigt oben die Verteilung der Gesamtimpulse der betrachteten Systeme aus Abschnitt 3.6.5. Der Gesamtimpuls ist in keinem der Systeme erhalten, was auch nicht zu erwarten ist. Auffällig ist jedoch die starke Abweichung der Verteilungen untereinander. Der Grund für diese Abweichung wird durch die mittlere und die untere Abbildung offenbar. Diese zeigen ebenfalls Verteilungen der Gesamtimpulse, allerdings aufgespalten in die horizontale Komponente (Mitte) und die vertikale Komponente (unten). Für die beiden ersten Systeme (right und down) kann den Abbildungen entnommen werden, dass die jeweils zu Beginn verschwindende Impulskomponente erhalten bleibt.

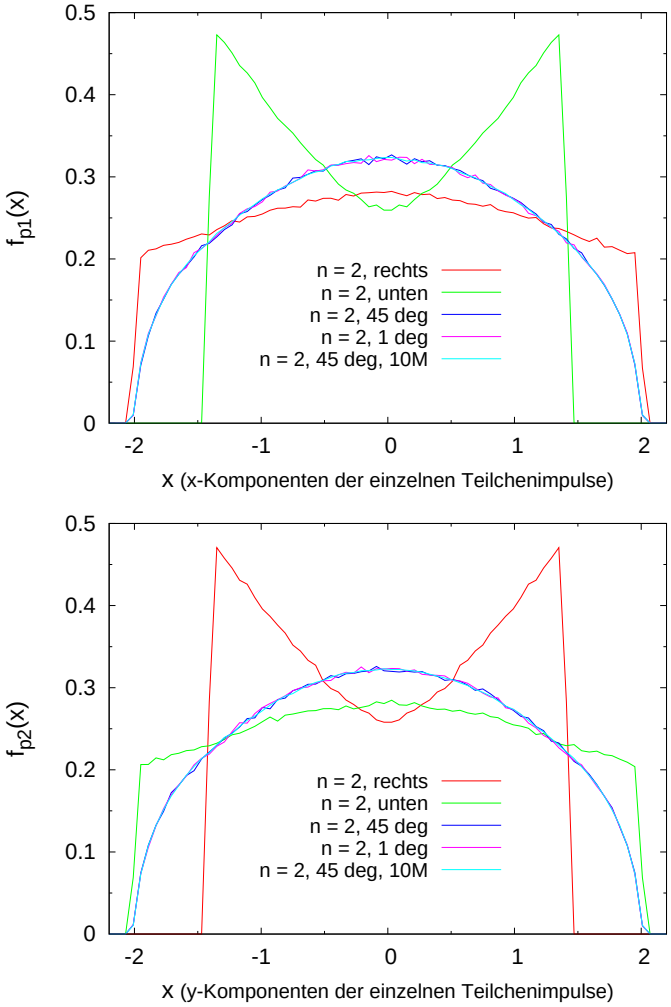


Abbildung 3.30: Diese Abbildung zeigt die Verteilung der Geschwindigkeitskomponenten aufgespalten nach den verschiedenen Raumrichtungen. Oben dargestellt sind die Verteilungen für die $x-$Koordinatenrichtung und unten die entsprechenden Verteilungen für die $y-$Richtung. Die Verteilung für die Teilchengeschwindigkeiten aus Abb. 3.28 unten scheint offenbar aus diesen beiden Verteilungen zusammengesetzt, was auch die anfänglich eher befremdliche Form der Verteilungen für die ersten beiden Systeme erklärt.

3.7 Diskussion

Die entwickelten Gleichungen (3.3, 3.5, 3.7) geben für alle getesteten Systeme den Verlauf der berechneten Verteilungen gut wieder, wie es aus den Abbn. 3.5 bis 3.8 in Abschnitt 3.5 ersichtlich ist. Die beobachteten Abweichungen, wie sie in den Abschnitten 3.6.1, 3.6.2, 3.6.3, 3.6.4 und 3.6.5 näher betrachtet wurden, sind hingegen noch nicht verstanden und können unterschiedliche Gründe haben. Auch wenn die Ergebnisse beider Molekulardynamik-Simulationsprogramme übereinstimmen kann ein Programmierfehler natürlich nie ausgeschlossen werden. Allerdings müsste ein solcher Fehler in der prinzipiellen Vorgehensweise der Simulationen liegen, da sich beide Programme stark voneinander unterscheiden. Der Fehler müsste weiterhin außerordentlich subtil in Erscheinung treten, da auch mit einer visuellen Inspektion keine Anomalie während eines animierten Simulationslaufs festzustellen ist.

Eine mögliche Fehlerquelle wären Rundungsungenauigkeiten, wie sie in Computersimulationen immer auftreten, jedoch erscheint dies eher fraglich, da sich der Effekt präzise auch für verschiedene Systeme mit gleichem N reproduzieren lässt, siehe dazu z. B. Abschnitt 3.6.1.

Bei den hartwandigen Systemen fällt auf, dass Abweichungen lediglich für $N = 2$ auftreten und zwar sowohl für d = 2 als auch für d = 3. Außerdem ist der Effekt bei allen drei betrachteten Verteilungen beobachtbar, siehe Abbn. 3.5 und 3.6. Für d = 3 ist die Abweichung bei der Verteilung der Geschwindigkeitskomponenten nur schlecht erkennbar, allerdings ist sie für d = 2 noch sehr deutlich.

Anders verhält es sich für die Systeme mit PR. Hier können Abweichungen für alle $N > 2$ für d = 2 und d = 3 beobachtet werden, wobei der Effekt mit zunehmendem N abnimmt, bis er schließlich nicht mehr wahrgenommen werden kann, siehe Abbn. 3.7 und 3.8. Die Fälle $N = 2$ fallen an dieser Stelle heraus, da sie $\delta-$förmig sind und somit von ihrer Form nicht abweichen können. Ein wichtiger Unterschied zu den Abweichungen für die hartwandigen Systeme ist die Tatsache, dass bei den Systemen mit PR die Verteilungen der Geschwindigkeitskomponenten unbeeinflusst zu bleiben scheinen. Eine Erklärung hierfür in Abhängigkeiten der Geschwindigkeitskomponenten untereinander liegen, deren Verteilung an sich zwar mit der Theorie übereinstimmen, nicht jedoch in Form der Beträge $v_i^2 = \sum_{\alpha=1}^{d} v_{i\alpha}^2$ oder der Energien $E_i = 1/2 \sum_{\alpha=1}^{d} v_{i\alpha}^2$. Dies mag als Hinweis gelten, dass sowohl die Beträge, als auch die Energien nicht als Funktion unabhängiger Variablen $v_{i\alpha}$ betrachtet werden können. Eine Antwort auf diese Frage können die Auto- und Kreuzkorrelationen der $v_{i\alpha}$ geben. Diese sollen im weiteren Verlauf der Untersuchungen berechnet werden.

Kapitel 4

Molekulargraphik von *coarse-grained*-Flüssigkeiten

4.1 Motivation

Ein Problem vieler mit *coarse graining* simulierenden Arbeitsgruppen ist es, die berechneten Systeme von Molekülen graphisch darzustellen. Tatsächlich existiert eine große Zahl an Molekulargraphikprogrammen. Als einige kostenfreie Beispiele seien folgende genannt: MolScript,[43] VMD,[44] Raster3D,[45] Chimera,[46] AtomEye,[47] RasMol,[48] gOpenMol,[49] Jmol,[50] PyMOL[51] und Molekel.[52] Einige kommerziell erhältliche sind Discovery Studio,[53] SYBYL[54] und MOLCAD.[55] Jedoch basieren diese im Wesentlichen auf der Darstellung atomistischer Modelle. Meist werden dann Atome als Kugeln oder Eckpunkte in einer „Neonröhren"-Graphik dargestellt.

Im Bereich der *coarse-grained*-Molekulardynamiksimulation werden ganze Moleküle oder Teile davon vereinfachend zu „groben" Einheiten zusammengefasst. Üblicherweise verwendete Einheiten sind gestreckte/gestauchte Kugeln (prolate/oblate/biaxiale Ellipsoide), mit Zylinder verbundene Kugelhälften (Spherozylinder), symmetrisch geschnittene Kugeln (Cut Spheres) oder geschlossene Tori (Spheroplatelets); Beispiele für solche Körper sind in Abb. 4.1 dargestellt. Auch etwas exotischere Definitionen kommen zum Einsatz; zum Beispiel der Kugelabschnitt, welcher in seiner Form etwa mit einer Kontaktlinse vergleichbar ist. Ein Beispiel für diesen Körper zeigt ebenfalls Abb. 4.1.

Die Familie der anisotropen Gay-Berne-Potentiale beispielsweise benutzt weiche Ellipsoide zur Repräsentation ganzer prolater[57] oder oblater[58] (und entsprechend meist mesogener) Moleküle um Monte Carlo- und Molekulardynamikberechnungen[59] durch Aufgabe der intramolekularen Struktur zu beschleunigen. Beispiele für die Verwendung einiger der anderen Modelle sind: Weiche Spherozylinder;[60] weiche biaxiale Ellipsoide;[61] harte Ellipsoide und Spherozylinder[62] und weitere.[63] Die Verwendung solcher nicht sphärischer meist konvexer starrer Körper wurde traditionell hauptsächlich in der Flüssigkristallforschung

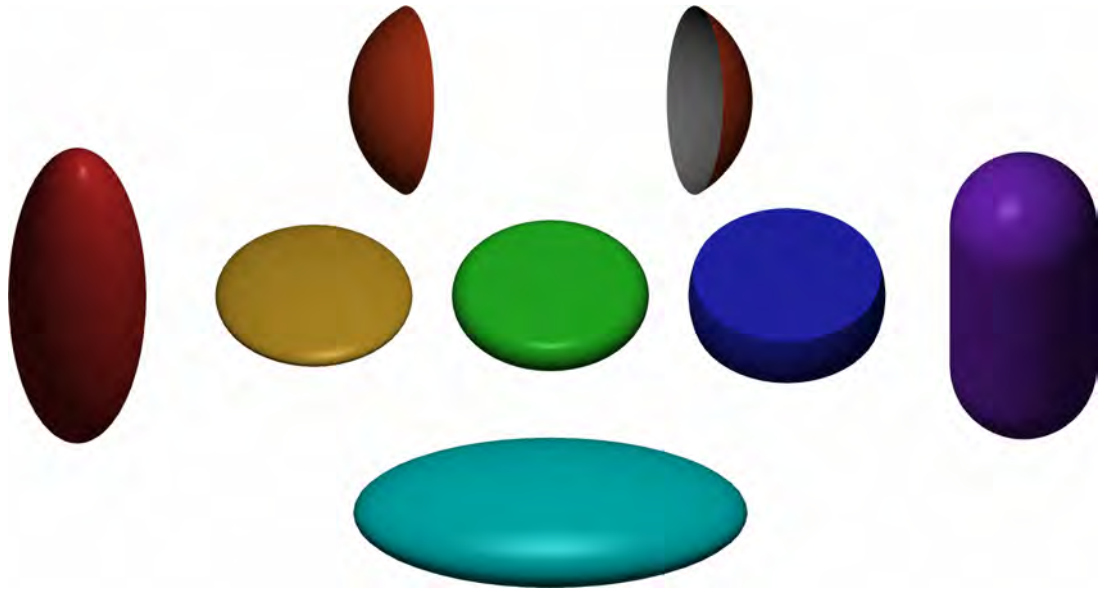


Abbildung 4.1: Die häufigsten Modelle zur Vereinfachung molekularer Strukturen bzw. ganzer Moleküle in der Molekulardynamik. Obere Zeile: Ein Beispiel für ein etwas ungewöhnlicheres Modell stellt der (Hohl)Kugelabschnitt dar. Dennoch findet auch diese Form Verwendung.[56] Die Integration in QMGA verdankt der Kugelabschnitt der Anfrage von Giorgio Cinacchi (School of Chemistry, University of Bristol, UK). Mittlere Zeile, von rechts nach links: Prolater Ellipsoid, Oblater Ellipsoid, Spheroplatelet, Cut Sphere, Spherozylinder. Untere Zeile: Beispiel für ein biaxiales Ellipsoid.

eingesetzt,[64–67] hat sich jedoch mittlerweile auch auf anderen Gebieten bewährt wie zum Beispiel in der mesoskopischen Beschreibung von Polymeren[65] oder, allgemeiner, für starre Einheiten innerhalb größerer Moleküle,[65,68] insbesondere bei Biomakromolekülen.[69]

Die meisten Molekulargraphikprogramme sind in der Lage, bestimmte Anordnungen von Atomen durch eine wachsende Anzahl an stilisierten Repräsentationen hervorzuheben, zum Beispiel die Sekundärstruktur eines Proteins als gewundenes Band (Ribbon),[70,71] molekulare Oberflächen,[72–75] Elektronendichte-Darstellungen und weitere. Jedoch werden Modelle, wie sie in der *coarse-grained*-Molekulardynamik notwendig sind, nicht direkt unterstützt. Außerdem lesen Standardprogramme üblicherweise lediglich Sets kartesischer Koordinaten für die einzelnen Atome $\{\vec{r}_i\}$ ein, nicht jedoch Orientierungen $\{\vec{e}_i\}$ wie sie zur korrekten Darstellung nicht kugelsymmetrischer Körper notwendig sind. Zwar finden abgeflachte, biaxiale Ellipsoide zum Beispiel als stilisierte Darstellung der Atome von DNS-Basen Verwendung. Jedoch beruht auch an dieser Stelle die Form und die Ausrichtung der Ellipsoide auf den kartesischen Koordinaten der eingelesenen Atome, aus welchen sie berechnet werden. Ein Versuch von unserer Seite, diese Funktionalität mit Hilfe präparierter Eingabedateien zur Anzeige mesogener Phasen zu verwenden, verlief nicht zufriedenstellend. Der Versuch bestand darin, eine MD-Konfigurationsdatei in das PDB-Dateiformat (Protein Data Base) zu konvertieren, wobei alle Ellipsoiden durch DNS-Basen, also deren zugrundeliegende Atome, dargestellt wurden, und diese Datei dann mit der bekannten

Visualisierungssoftware Chimera zu laden. Leider erwies sich Chimera, welches auf dem Gebiet der atomistischen Darstellung Hervorragendes leistet, als nicht geeignet für diesen Zweck. Kleinere Konfigurationen von einigen tausend Ellipsoiden konnten noch problemlos geladen werden, doch bei für Flüssigkristallsimulationen typischen Größen von einigen zehn- oder gar hunderttausend Teilchen fror das Programm dauerhaft (mehr als einige Stunden) ein und wurde unbenutzbar. Hinzu kommt die Tatsache, dass jeder Ellipsoid durch die Vielzahl von Koordinaten aller seiner ihn repräsentierenden Atome dargestellt wird, was die Dateigröße erheblich steigert und unhandlich macht.

Eine graphische Darstellung der berechneten Systeme ist aber in vielerlei Hinsicht nützlich, wenn nicht gar notwendig. Einerseits hilft ein anschauliches, drehbares Bild bei der Analyse der Berechnungen, sowohl für einen ersten Gesamteindruck, als auch beim Finden auffälliger Strukturen. Andererseits sind Bilder für jegliche Art der Präsentation, wie zum Beispiel Veröffentlichungen, Vorträge und Poster nahezu unabdingbar.

Die Folge ist, dass viele Wissenschaftler, die auf diesem Gebiet tätig sind, auf eigene Lösungen für dieses Problem zurückgreifen müssen,[76,77] oder Programme verwenden, die für andere Zwecke gedacht sind,[78,79] siehe zum Beispiel unser zuvor beschriebener Versuch. Einige solcher Behelfslösungen sind, abgesehen von unbequem und zeitintensiv, auch nicht in der Lage visuell mit dem Benutzer rückzukoppeln, bevor das Bild vollendet ist, also zum Beispiel freies Drehen und Zoomen sind nicht möglich. Als Beispiel für eine solche Prozedur sei eine Reihe von Shell- und Perl-Skripten genannt, mit deren Hilfe aus einer Konfigurationsdatei ein Pov-Ray Skript erzeugt werden kann, um mit diesem ein einzelnes Bild zu Rendern. Diese Herangehensweise ist offenbar kompliziert, zeitintensiv und gibt dem Benutzer keinerlei Rückmeldung oder Möglichkeit zur Interaktion bis das fertige Bild erzeugt ist.

Im Rahmen dieser Arbeit wurde diese Lücke mit einer OpenGL[80] basierten Molekulargrafikapplikation geschlossen, welche unter GPL (Gnu Public Licence) Lizenz als Quellcode frei verfügbar ist.[81,82] Der Name des Programms ist QMGA, ein Akronym für Qt-based Molecular Graphics Application. Ein Bildschirmfoto des Hauptfensters, welches ein Testsystem zeigt, ist in Abb. 4.2 zu sehen. Wir haben uns entschlossen ein komplett neues Programm zu schreiben, anstatt ein vorhandenes zu modifizieren, da letzteres von Hause aus über eine große Menge an, typischerweise biomolekularer, Funktionalität verfügt, welche für unsere Zwecke nicht notwendig ist und die Entwicklung unnötigerweise verkompliziert. Auf diese Weise waren wir auch in der Lage speziell auf die Anforderungen an eine Software zur Flüssigkristallvisualisierung einzugehen, also zum Beispiel auch die nötige Performanz um Systeme von $10^4 - 10^5$ Objekten darzustellen, wie sie auf diesem Gebiet üblich sind.

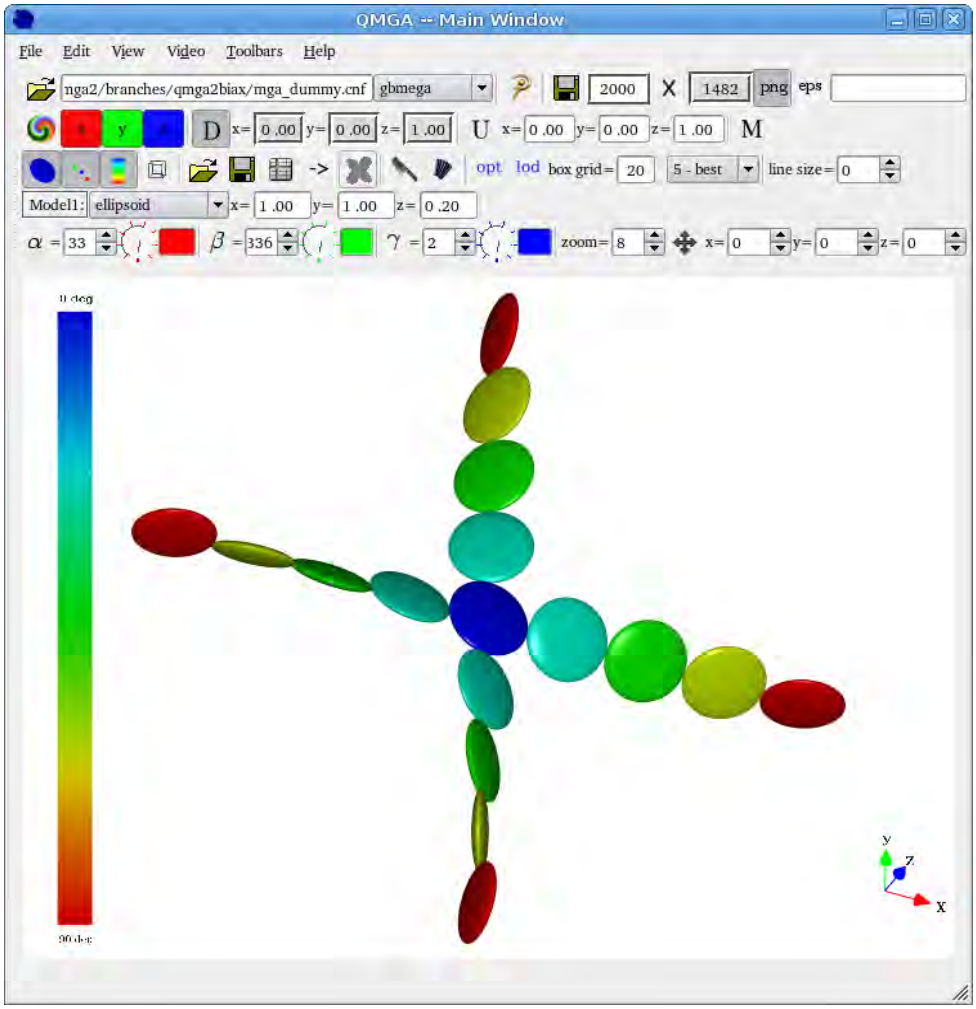
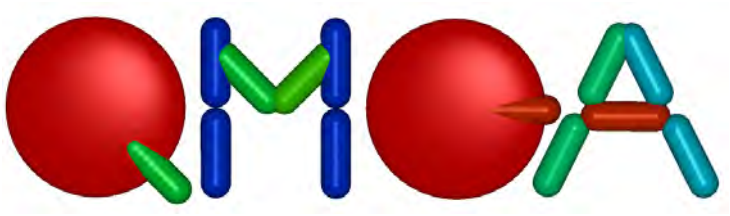


Abbildung 4.2: Bildschirmfoto von QMGA. Dargestellt wird ein künstlich erzeugtes Testbild zur Veranschaulichung der Winkelabhängigkeit der Farbgebung. Der Abbildung kann entnommen werden, dass es sich bei der derzeitigen Bezugsachse um die z-Achse handelt. Erkennbar ist im Vergleich mit der Farbpalette ebenfalls, dass die Farben Blau und Rot für Ausrichtungen parallel beziehungsweise senkrecht zur Bezugsachse stehen.



4.2 Programmfeatures

Eine Liste von Anforderungen, die unserer Meinung nach an ein solches Programm gestellt werden müssen, ist:

- Vollständig gerenderte Darstellung des Systems
- Vereinfachte Darstellung mit Richtungsvektoren (Stick View)
- Farbkodierung der Modelle (Color Coding)
- Zoom und Rotation
- Tastatur- und Mausinterface
- Screenshot-Funktionalität
- Schneiden des Systems (Slicing)
- Video-Funktionalität
- Darstellung von Gemischen verschiedener Modelle
- Falten von Systemen mit periodischen Randbedingungen
- Lichtoptionen
- Einstellbare Renderqualität
- Speichern von Optionen
- Darstellung biaxialer Modelle

Die gegebene Reihenfolge gibt grob die Wichtigkeit der entsprechenden Funktionen an. Alle diese Anforderungen und viele mehr werden von QMGA erfüllt und einige im Folgenden näher beleuchtet.

4.2.1 Gerenderte und vereinfachte Darstellung

Ein gerenderte Ansicht ist sicherlich die absolute Basis, ansonsten gäbe es nichts zu sehen. Vollständiges Rendern soll bedeuten, dass jedes Modell als ein ausgefüllter, dreidimensionaler konvexer Körper dargestellt wird. Dieser Darstellung liegt ein Set von Dreiecken (Wireframe) zugrunde, welche die gewünschte Form nähern; vgl. Abb. 4.3. Dem gegenüber steht die Darstellung lediglich des Orientierungsvektors eines jeden Moleküls; vgl. Abb. 4.4. Letztere Ansicht liefert zwei Vorteile. Einerseits ist es somit möglich, auch durch sehr dichte Systeme hindurchzuschauen und so Strukturen im Inneren wahrzunehmen; andererseits ist diese Ansicht viel weniger rechenintensiv und kann somit, insbesondere auf langsameren Computern, benutzt werden, um das Zoomen und Rotieren großer Systeme zu beschleunigen.

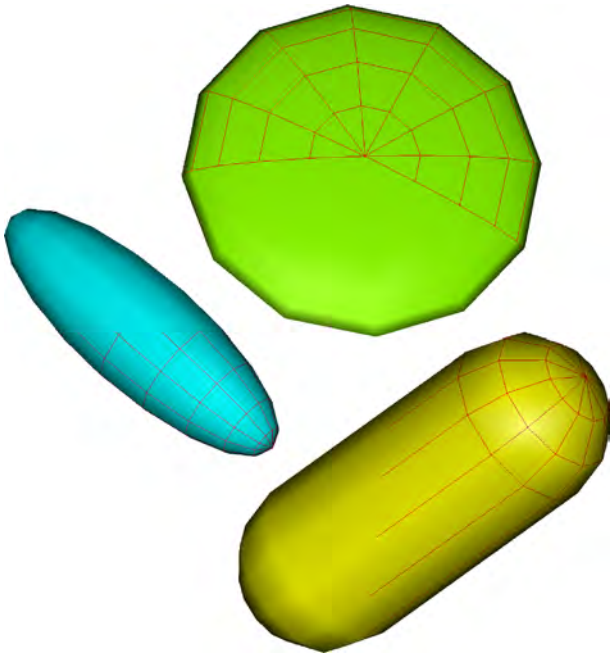


Abbildung 4.3: Oblater (oben) und prolater (links) Ellipsoid, sowie ein Spherozylinder (unten). Den Modellen wurde teilweise das zugrundeliegende Wireframe überlagert, um die Polygonstruktur zu verdeutlichen. Gezeigt ist die Struktur für eine mittlere Einstellung der Renderqualität. Eine komplette Darstellung der voreingestellten Renderqualitäten findet sich in Abb. 4.11.

4.2.2 Farbkodierung

In konventionellen Molekulargraphikprogrammen sind oft Atome repräsentierende Kugeln die primären Modelle, zumindest in einer „Ball and Stick“ Ansicht. Es sei erwähnt, dass es auch eine Anzahl anderer Darstellungen gibt (z.B. Neonröhren). Sowohl bei der Darstellung mit Kugeln, wie auch mit Neonröhren, werden Atome häufig nach dem Corey-Pauling-Koltun Schema gefärbt: Weiß für Wasserstoff, schwarz oder grau für Kohlenstoff, blau für Stickstoff, rot für Sauerstoff, und so fort.[48,84,85]

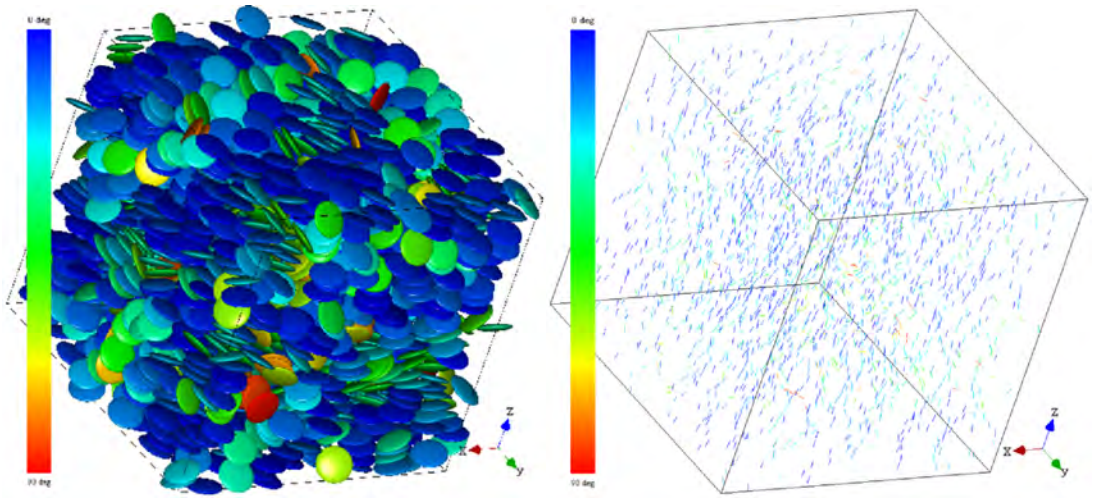


Abbildung 4.4: Von weichen, oblaten Ellipsoiden gebildete nematische Phase. Die Wechselwirkungen werden über das GBDII-Potential[58] modelliert ($\mu = 1$, $\nu = 2$, $\kappa = 0.2$, $\kappa' = 0.1$, $T^* = 12$, $P^* = 200$).[83] Vollständig gerenderte Darstellung links und vereinfachte Darstellung rechts.

Andere Schemata zur Farbgebung basieren auf Eigenschaften wie Hydrophobizität, Ladung, dem Betrag der Geschwindigkeit $v_i = |\vec{v}_i|$ und entsprechend, wegen des Gleichverteilungssatzes, der Temperatur $T_i = 3m_i v_i^2/k_\mathrm{B}$, aber zum Beispiel auch auf der Orientierung $\vec{e}_i$ nicht kugelsymmetrischer Körper. Gerade letzteres ist im Bereich der mesoskopischen Systeme außerordentlich nützlich, um einen ersten Eindruck der Gesamtordnung der Phase zu erhalten, da bei geordneten Phasen eine bestimmte Farbe überwiegt. Diese Art der Darstellung ist mindestens seit den frühen 1990er Jahren in Gebrauch.[76,77] Jedem Molekül i wird dabei eine Farbe in Abhängigkeit von $c_i = |\vec{e}_i \cdot \vec{n}| \in [0, 1]$ gegeben, also dem Betrag des Skalarproduktes zwischen dem molekülfesten Richtungsvektor $\vec{e}_i$ und einem Einheitsvektor $\vec{n}$, der systemfest definiert und somit für alle Moleküle gleich ist. $\vec{n}$ kann dabei zum Beispiel der für Mesophasen üblicherweise definierte, über die Box gemittelte Direktor sein, eine der drei kartesischen Raumrichtungen $\vec{e}_x, \vec{e}_y, \vec{e}_z$, oder auch eine bestimmte Symmetrieachse des Systems. Letzteres ist zum Beispiel in Abb. 4.7 zu sehen, wo $\vec{n}$ die Rotationsachse eines zylindrischen Systems ist.

Der Direktor ist definiert als der Eigenvektor zum betraglich größten Eigenwert des Ordnungstensors $\hat{Q}$:

$$\hat{Q} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{3}{2}\vec{e}_i \otimes \vec{e}_i - \frac{1}{2}\right). \tag{4.1}$$

Generell bezeichnet Color Coding eine Verbindung eines variablen Wertes $x \in [a, b] \subset \mathbb{R}$ einer relevanten Eigenschaft mit einer Farbpalette M, im Allgemeinen einer diskreten Liste von Farben. Die Zuordnung wird mittels einer Funktion der Form $f_\mathrm{col}(x) : [a, b] \mapsto M$ realisiert. Die explizite Form von f_col und M hängen stark von der tatsächlichen Anwendung ab. Im vorliegenden Fall wird $x = c_i$ einer diskreten, regenbogenartigen Farbpalette

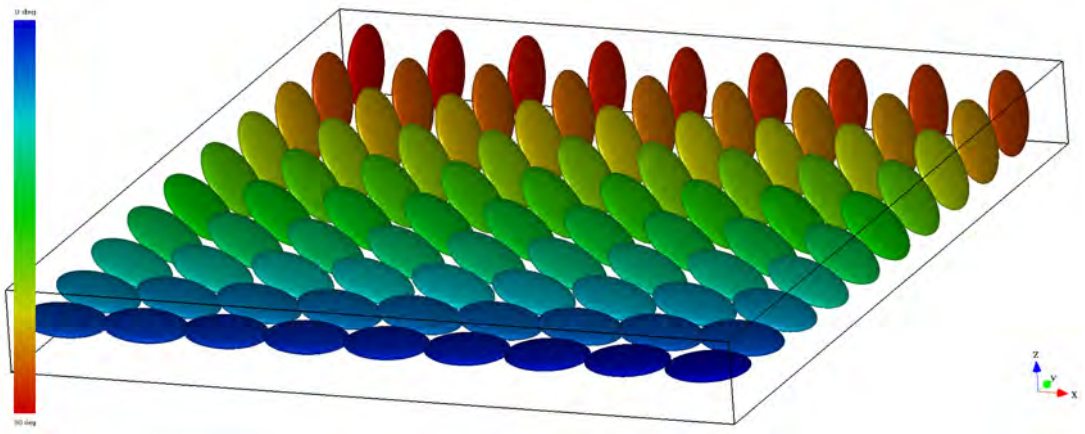


Abbildung 4.5: Visualisierung der implementierten Standardpalette von 91 Farben für die Winkel zwischen Referenzachse und Molekülachse im Bereich $[0, 90]$. Die Referenzachse ist in diesem Beispiel die z-Achse des Systems.

M_{RGB} von RGB-Werten (Anteile von R=rot, G=grün und B=blau) zugeordnet. M_{RGB} an sich besteht aus 91 verschiedenen Farben, eine für jedes Grad von $\arccos c_i \in [0, 90]$, wobei jede Farbe durch ein Tupel dreier Werte $R, G, B \in \mathbb{N}$ im Intervall $[0, 255]$ dargestellt wird. Eine Darstellung der benutzten Farbpalette zeigt Abb. 4.5. Sowohl der Direktor, als auch ein vom Benutzer definierbarer Vektor werden in der GUI angezeigt. Letzterer kann zur Programmlaufzeit beliebig verändert werden, wobei der Effekt auf die Färbung mit sofortiger Wirkung eintritt.

Alternativ zur Farbgebung in Abhängigkeit zur Ausrichtung können bestimmte Modelle mit einer einheitlichen Farbe belegt werden. Nützlich ist dieses Feature beispielsweise um die verschiedenen Arten eines Gemisches farblich voneinander zu trennen, siehe Abb. 4.8, oder um ein einzelnes Molekül hervorzuheben.

4.2.3 Benutzer-Interface

Die 3D-Darstellung des Systems kann mit der Maus frei gedreht und gezoomt werden. Informationen zur aktuellen Position werden in Form von Winkeln und Zoomfaktor angegeben und beschreiben den Raumpunkt von dem aus der Benutzer das System betrachtet. Alle Werte werden kontinuierlich angepasst, wenn das System gedreht oder gezoomt wird. Es ist ebenfalls möglich, diese Werte über die Tastatur einzugeben, so dass die Ansicht im Renderbereich angepasst wird. Auf diesem Weg ist es möglich, einen bestimmten Blickwinkel exakt zu reproduzieren, beispielsweise um zwei Systeme miteinander zu vergleichen.

4.2.4 Bildschirmfotos

Ein Grafikprogramm ist nicht nur geeignet, um die Ergebnisse der eigenen Arbeit besser zu verstehen, sondern auch um diese der Öffentlichkeit zu präsentieren. Hierfür sind aber

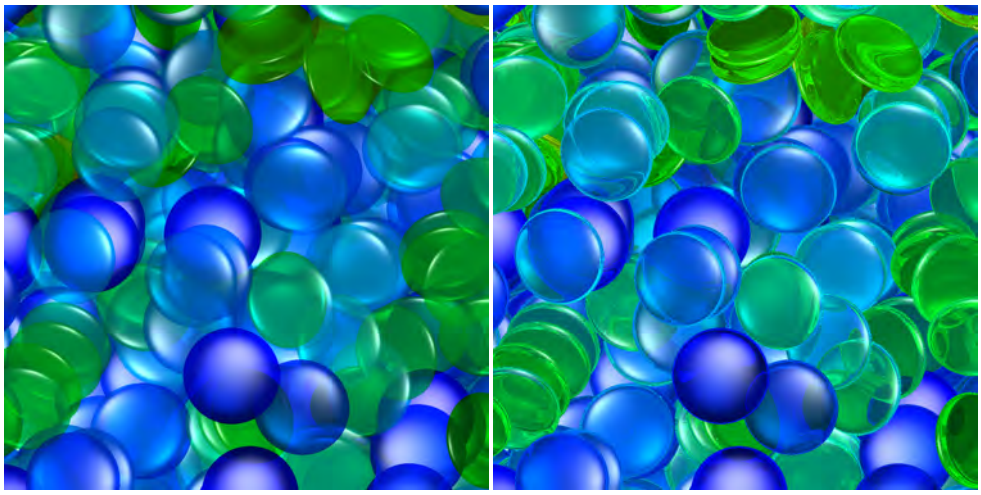


Abbildung 4.6: Die linke Abbildung zeigt eine Nahaufnahme einer nematischen Phase oblater Ellipsoide und wurde mit der POV-Ray-Exportfunktionalität von QMGA erzeugt. Die Darstellung der Objekte als nicht lichtbrechende durchsichtige Körper ist eines der von QMGA angebotenen Templates. Dem Benutzer ist es jedoch selbstverständlich freigestellt, die Templates jederzeit seinen Bedürfnissen entsprechend anzupassen. Beispielsweise ist es durch Einkommentieren einer vordefinierten Zeile im generierten Skript möglich, die Lichtbrechung der Körper mit einem frei wählbaren Brechungsindex n zu aktivieren. Die rechte Abbildung zeigt das Ergebnis einer Wahl von $n = 1.5$ für den Brechungsindex.

Bilddateien absolut notwendig und damit auch die Möglichkeit, diese aus der Renderfläche zu erzeugen, also Bildschirmfotos von letzterer zu generieren. Bei praktisch allen Programmen zum direkten Speichern von Bildschirmfotos ist die Auflösung des Bildschirmfotos abhängig von der Größe des dargestellten Fensters und somit von der Auflösung des Monitors. Daraus resultiert, dass es nicht möglich ist, Bilder mit einer Auflösung zu generieren, die höher ist als die des Monitors. Dies wiederum führt zu Problemen (unscharf, pixelig), wenn Bilder in großem Format, beispielsweise auf Konferenzpostern, gedruckt werden sollen.

QMGA erlaubt dem Benutzer, die gewünschte Auflösung unabhängig vom Ausgabegerät zu wählen, was insbesondere für sehr große Ausdrucke nützlich ist, aber auch zum Erzeugen stark zoombarer Bilddateien. Als Ausgabeformate werden PostScript und PNG unterstützt. Das Seitenverhältnis wird im Übrigen automatisch richtig angepasst, so dass auch bei Änderungen der gewünschten Auflösung oder Größenänderungen des Programmfensters keine Deformationen auftreten. Es ist außerdem möglich, ein automatisch generiertes, aber anpassbares POV-Ray[79] Skript zu exportieren, um die Möglichkeiten, welches dieses mächtige Raytracingprogramm zur Verfügung stellt ausschöpfen zu können. Ein Beispiel für eine POV-Ray-Darstellung mit durchsichtigen Ellipsoiden ist in Abb. 4.6 zu sehen.

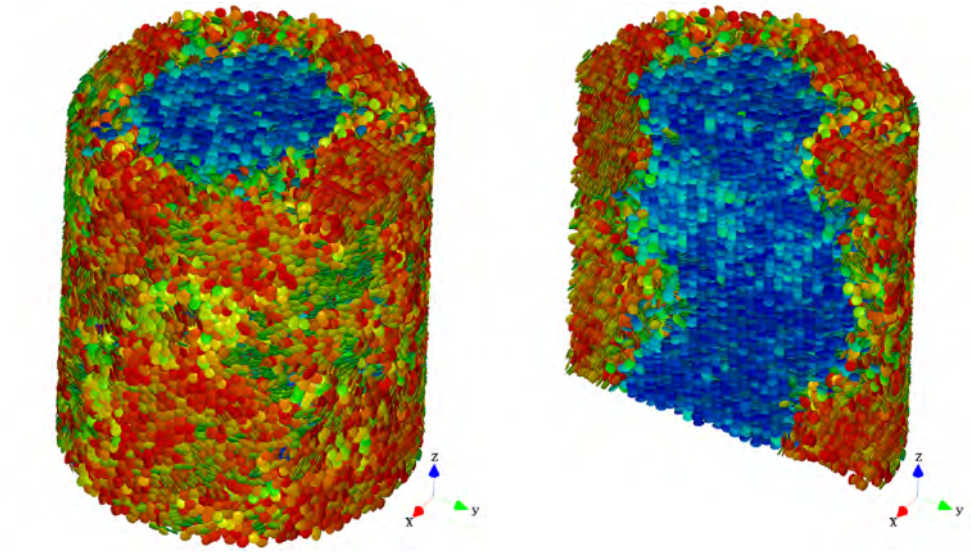


Abbildung 4.7: Schnappschuss einer Moleküldynamiksimulation eines diskotischen Mesogens, welches in einer zylindrischen Nanopore eingeschlossen ist.[86] Um zu zeigen, dass die Simulation in der Mitte eine kolumnar geordnete Phase ausbildet, welche von einer schwach geordneten Schale umgeben ist, wurde das System in der Mitte entlang der Zylinderachse geschnitten, wobei die Slice-Funktion von QMGA zum Einsatz kam.

4.2.5 Schnitte

Bei einem dicht gepackten System ist das Innere nicht zu sehen. Bisweilen kommt es jedoch vor, dass genau das Innere des Systems die interessante Region darstellt. Zwar kann mit der vereinfachten Darstellung über die Richtungsvektoren das System durchblickt werden, dennoch ist für viele Details eine vollständig gerenderte Darstellung notwendig. Im letzteren Fall muss das System geschnitten werden. Ein Beispiel: Abb. 4.7 zeigt die Simulation eines diskotischen* Flüssigkristalls innerhalb einer Nanopore.[86] In der Abbildung wurde ein Schnitt entlang der Porenachse gelegt. Beim Schneiden entscheidet QMGA aufgrund der Lage des Objektmittelpunkts, ob dieses dargestellt wird oder nicht. Das bedeutet, dass keine Objekte geschnitten werden: Entweder sie sind vollständig zu sehen oder vollständig verborgen.

4.2.6 Videos

Da molekulare Simulationen üblicherweise in der Zeit fortschreiten, ist die Möglichkeit bewegte Bilder darzustellen und aufzuzeichnen von Vorteil. QMGA ist in der Lage, eine Sequenz von Konfigurationsdateien zu laden und nacheinander anzuzeigen. Auf diesem Weg wird der Eindruck eines bewegten Bildes erzeugt. Das Interface erlaubt die üblichen Funktionen, welche man von einem Video-Abspieler erwartet, namentlich Start, Stopp und Pause. Zusätzlich sind noch vor- und rückwärts Abspielen implementiert, sowie auch

*Die Moleküle werden durch oblate Ellipsoide dargestellt. Der Name entstammt der Ähnlichkeit zu einem Diskus.

eine Capture-Option.

Bei sehr großen Systemen und langsamen Rechnern nehmen die Lade- und Renderzeiten zu, bis irgendwann die Darstellung zu langsam wird, als das sie noch benutzbar wäre. In einem solchen Fall ist es möglich und auch ratsam über die Captureoption alle angezeigten Einzelbilder als Bilddateien auf der Festplatte zu speichern und aus diesen ein echtes Videofile zu generieren. Zwar kann die Speicherung aller Bilder von QMGA automatisch erledigt werden, aber zur Zeit existiert noch keine Möglichkeit, diese automatisch zu einer Videodatei zu kodieren. Zu diesem Zweck muss ein externes Programm benutzt werden. Gute und kostenfreie Lösungen stellen die Programme ffmpeg[87] und avidemux[88] dar. Letzteres verfügt sogar über eine einfach zu bedienende, graphische Benutzeroberfläche.

4.2.7 Gemische

Obwohl viele (*coarse-grained*) Molekülsimulationen reine Phasen behandeln, gibt es auch solche mit mehr als einer Art von Molekülen. Um diesem Umstand gerecht zu werden, beinhaltet die intern von QMGA benutzte Molekülklasse eine Variable von ganzzahligem Wert, welcher für jedes Molekül einzeln gesetzt werden kann. Diese Variable wird als ein Etikett benutzt, um sämtliche Moleküle in Gruppen aufzuteilen. Im Weiteren ist es möglich, jeder Gruppe ihre eigenen Modellparameter zuzuordnen. In einem extremen Fall wäre es möglich, jedem einzelnen Molekül ein eigenes Etikett und damit ein eigenes Aussehen zu geben.

Aus Gründen der Benutzerfreundlichkeit werden zwei Werkzeugleisten direkt im Hauptfenster von QMGA angezeigt, über welche die Parameter der ersten zwei Modelle eingestellt werden können. Zu viele Leisten verschlechtern die Übersicht eher, als ihr zu nützen, und die meisten derzeit üblichen Simulationen gebrauchen ohnehin nicht mehr als zwei Molekülsorten; für einige Beispiele siehe Abb. 4.8. Aus diesem Grund wurde von der Implementierung weiterer Werkzeugleisten zum Einstellen der Modellparameter direkt im Hauptfenster abgesehen. Statt dessen können die Parameter aller Modelle übersichtlich in einer Liste angeordnet in einem eigenen Fenster geöffnet und editiert werden, siehe Abb. 4.9.

4.2.8 Periodische Randbedingungen

Die meisten Molekülsimulationen machen Gebrauch von periodischen Randbedingungen (PR). Die Darstellung eines solchen Systems kann grundsätzlich auf zweierlei Weisen geschehen. Einerseits können die Positionen aller Teilchen innerhalb der zugrunde liegenden Einheitsbox dargestellt werden, andererseits ist es auch möglich, die absoluten Positionen zu verwenden, welche sich ohne die Anwendung der PRn ergeben.

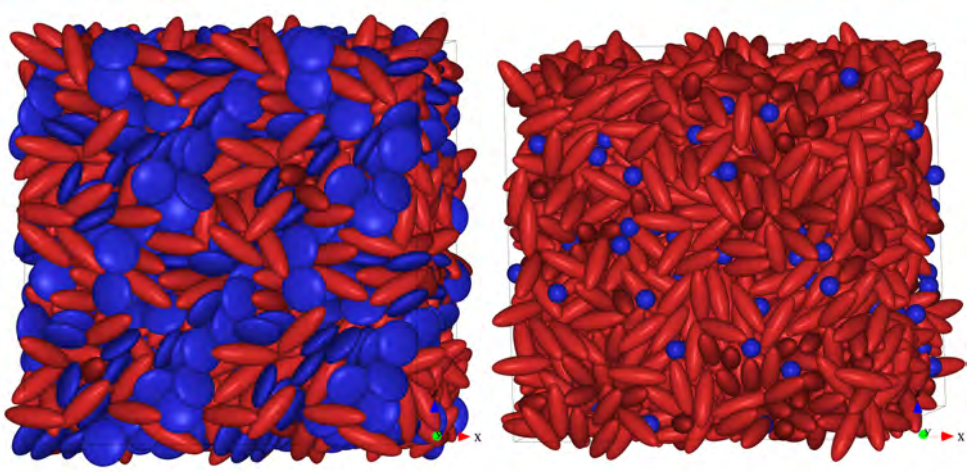


Abbildung 4.8: Beispiele zur farblichen Trennung verschiedener Modelle bei gemischten Systemen. Die abgebildeten Systeme wurden freundlicherweise von David Cheung der Molecular Simulation Group von Prof. Michael P. Allen (Department of Physics, University of Warwick, UK) zur Verfügung gestellt.

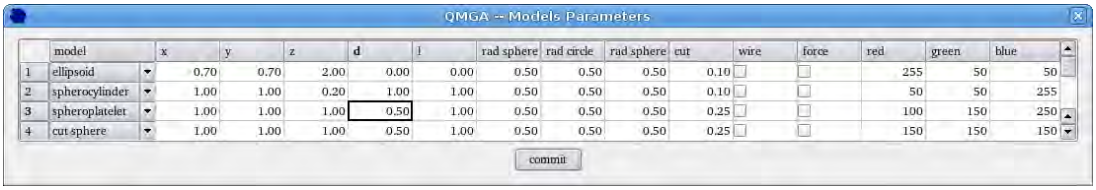


Abbildung 4.9: Bildschirmfoto des Dialoges, der alle Modellparameter in einer übersichtlichen und editierbaren Liste darstellt. Hier kann jederzeit auf sämtliche Parameter aller Modelle Einfluss genommen werden. Die Parameter der beiden ersten Modelle sind zusätzlich noch über zwei Werkzeugleisten des Hauptfensters zugänglich.

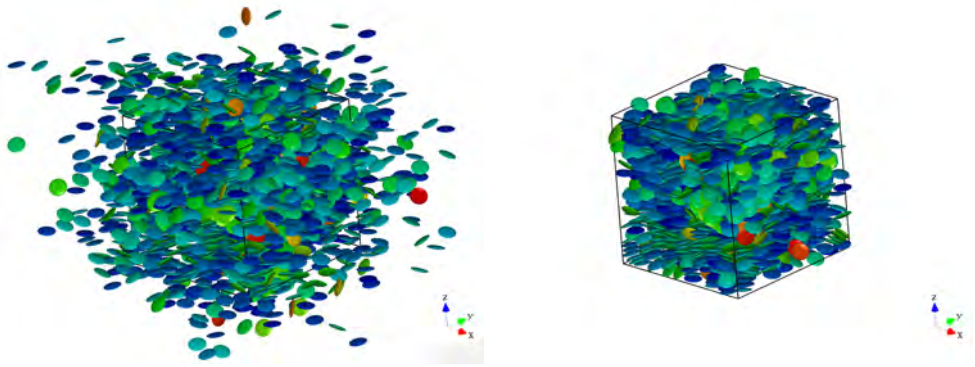


Abbildung 4.10: Das Falten einer Konfiguration tatsächlicher Teilchenpositionen zurück in die Simulationszelle ist bei periodischen Randbedingungen immer möglich, hier gezeigt am Beispiel einer nematischen Phase diskotischer Moleküle. Links ist die Darstellung der entfalteten Ausgangskonfiguration zu sehen und rechts die resultierende Konfiguration, nachdem alle Teilchenorte in die Ausgangszelle zurückgefaltet wurden.

Eine Faltung absoluter Koordinaten zurück in die Einheitszelle ist immer ohne Weiteres auch bei einer einzelnen Konfiguration möglich. Wenn aber die Koordinaten mit angewendeten PRn vorliegen, wird zum Entfalten eine komplette Trajektorie benötigt, welche mit einer Konfiguration beginnt, bei der sich alle Teilchen innerhalb der Einheitszelle befinden.[1] Im Folgenden wird das notwendige Vorgehen zum Entfalten einer solchen Trajektorie kurz erläutert. Zum Zeitschritt i sei mit r_i die gefaltete und mit $\overline{r}_i$ die tatsächliche Position eines Teilchens bezeichnet. Dabei steht r für eine beliebige der kartesischen Raumrichtungen x, y oder z. L_r benennt die Länge der Box in der entsprechenden Raumrichtung r. Mit folgender Rekursion ist es dann möglich, die tatsächliche Trajektorie des Teilchens zu rekonstruieren:

$$\overline{r}_0 = r_0 \tag{4.2}$$

$$\overline{r}_{i+1} = \overline{r}_i - \mathrm{Int}\left(\frac{r_{i+1} - r_i}{\frac{1}{2}L_r}\right) L_r \tag{4.3}$$

Die Startkonfiguration der entfalteten Trajektorie fällt mit dem Start der gefalteten zusammen. Nun wird jeweils, pro Raumrichtung, die Differenz zur folgenden Konfiguration bestimmt. Ist diese größer als eine halbe Boxlänge, so wurde eine Faltung vollzogen, und dementsprechend muss die tatsächliche Position um eine ganze Boxlänge korrigiert werden.

4.2.9 Licht- und Farboptionen

Zumeist ist es eine Frage des Geschmacks, wie die dargestellten Modelle hinsichtlich der Licht- und Oberflächenbeschaffenheit erscheinen sollen. OpenGL stellt die Funktionalität zur Verfügung, um den Objekten beispielsweise eine mattes oder glänzendes, eher metallisches oder plastikartiges Finish zu verleihen. Alle hierzu benötigten Parameter können

über ein Dialogfenster angepasst werden. Die Auswirkungen auf die Färbung werden dabei sofort wirksam. Die Farbe des Hintergrunds lässt sich frei einstellen, um der aktuellen Anwendung gerecht zu werden. Im Druck zum Beispiel wird in den meisten Fällen Weiß die beste Hintergrundfarbe sein, während am Bildschirm leicht gefärbt bis hin zu schwarz eher als angenehm in den Augen empfunden wird. Hinzu kommt natürlich auch der persönliche Geschmack des Benutzers.

4.2.10 Renderqualität

Die benötigte Zeit zum Rendern eines Systems hängt von einigen Parametern ab. Wichtig, aber vom Standpunkt des Programmierers nicht beeinflussbar, sind an dieser Stelle beispielsweise die Qualität des Grafikkartentreibers, sowie die Leistung der Grafikkarte und der CPU. Eine weitere wichtige und für den Programmierer beeinflussbare Rolle kommt der Gesamtzahl zu zeichnender Dreiecke zu. Während des Arbeitens ist flüssiges Drehen und Zoomen des Systems weit wichtiger als eine exzellente Darstellung der einzelnen Körper. Im Falle eines Bildschirmfotos ist es jedoch umgekehrt. Das resultierende Bild soll schön anzuschauen sein, aber bei einem stehenden Bild ist die Rendergeschwindigkeit nicht mehr von Belang. Um ein gewisses Maß an Einflussnahme zu gestatten, sind fünf Renderqualitäten von sehr grob bis nahezu perfekt gerundet voreingestellt und können bei Bedarf jederzeit gewechselt werden. Abb. 4.11 zeigt den Einfluss der Renderqualität am Beispiel eines einzelnen oblaten Ellipsoiden.

4.2.11 Remotezugriff auf Dateien

Molekülsimulationen werden häufig über das Netz auf Supercomputern berechnet, aber die Resultate sollen lokal auf dem Bildschirm des Anwenders dargestellt werden. QMGA vereinfacht den Datentransfer mit *ssh* um das Dateisystem des entfernten Rechners als Verzeichnisbaum darzustellen. Diesen kann der Anwender in von üblichen Dateibrowsern gewohnter Weise mit Maus oder Tastatur navigieren. Um eine bestimmte Datei in den Renderbereich zu laden, genügt beispielsweise ein Doppelklick mit der Maus und QMGA kopiert sie auf den lokalen Rechner. Die Datei wird in einem Verzeichnis für temporäre Daten abgelegt und geladen. QMGA zeigt ebenfalls an, wie viel temporäres Datenmaterial derzeit gespeichert wird, und bietet auch einen Knopf zum Leeren des temporären Verzeichnisses an. Das Einloggen auf dem entfernten Rechner und der Dateiaustausch sind über *ssh* und *scp* realisiert.

4.2.12 Speichern von Optionen

QMGA speichert bei Programmende eine Reihe (derzeit schon über hundert) von Optionen, um diese bei Programmstart wieder einzulesen. Dies sind beispielsweise, um nur

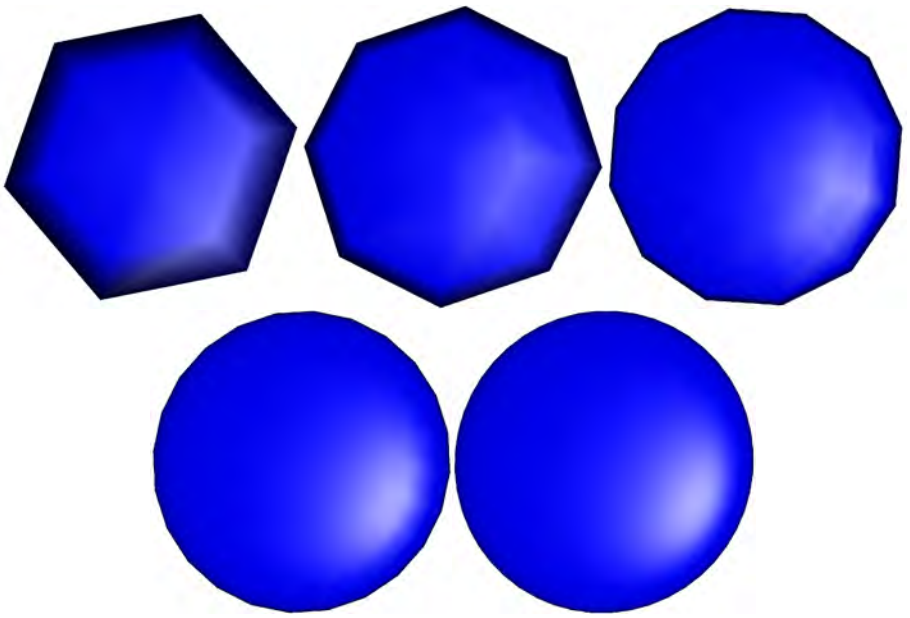


Abbildung 4.11: Diese Abbildung demonstriert den Effekt der Qualitätseinstellung des Renderns am Beispiel eines oblaten Ellipsoiden. Derzeit sind fünf vordefinierte Einstellungen zur Qualität in QMGA implementiert und decken den Bereich von sehr grob bis sehr gut vollständig ab. Niedrige Einstellungen ermöglichen es dem Benutzer, die Reaktion der Darstellung auf Drehen und Zoomen flüssiger zu gestalten, sollte dies notwendig werden. Direkt vor einem Bildschirmfoto sollte dann aber eine hohe Qualität gewählt werden, um den Ausdruck optisch ansprechend zu gestalten.

einige zu nennen:

- die letzte geöffnete Datei
- welche Werkzeugleisten wurden angezeigt, welche versteckt
- die Fensterposition
- der Rendermodus (voll oder vereinfacht)
- die Renderqualität
- eine kleine Historie geöffneter Dateien
- etc.

Außerdem kann es bei Systemen mit einer größeren Anzahl an verschiedenen Modellen recht mühselig sein, sämtliche Modellparameter für alle Modelle immer wieder von Hand einzugeben. Aus diesem Grund wurde es im Sinne der Benutzerfreundlichkeit ermöglicht, die gerade eingestellten Parameter in einer Textdatei zu speichern und natürlich auch wieder zu laden.

4.3 Programminterna

4.3.1 Die Programmstruktur

QMGA, wie auch die meisten verwendeten Bibliotheken, ist komplett in der Programmiersprache C++ mit einem objektorientierten Ansatz geschrieben. Die Fensterumgebung wurde mit Trolltechs Qt[89] realisiert, welches es erlaubt, selbst komplexe GUIs mit vertretbarem Aufwand zu erstellen. Für den 3D-Teil der Anwendung fiel die Entscheidung auf OpenGL[80] statt VRML[90] bzw. seinen Nachfolger X3D.[91] Die Entscheidung begründet sich zum einen auf der deutlich höheren Performance von OpenGL und zum anderen auf der Tatsache, dass VRML/X3D nie die Verbreitung und Unterstützung von OpenGL erreicht haben.

Die meisten Programme zur Molekulardynamik geben ihre Daten in Form von Textdateien aus, welche Informationen zu jedem einzelnen Molekül beinhalten. Es schien praktikabel, zu Beginn eine kleine Bibliothek von Objekten aufzusetzen, welche in der Lage ist, ein komplettes System von Molekülen einzulesen und im Hinblick auf Visualisierung zu behandeln. Diese Bibliothek besteht im Wesentlichen aus drei Klassen:

- Die Klasse *Molecule* modelliert ein einzelnes Molekül mit allen zugehörigen Parametern und Funktionen um ein solches zu beschreiben.
- Die Klasse *Colormap* liest eine Datei ein, welche RGB-basierte Farbwerte enthält. Des weiteren verfügt sie über die nötigen Funktionen, um den Molekülen diese Farbwerte nach bestimmbaren Regeln zuweisen zu können.
- Die Klasse CnfFile (Configuration File) liest alle notwendigen Daten aus der Ausgabedatei einer gegebenen Simulation ein. Das Herzstück dieser Klasse stellen ein Vektor aller *Molecule* Objekte und ein *Colormap* Objekt zu deren Farbgebung dar.

Die programmiertechnische Basis von QMGA bildet Qt3, mit dessen Hilfe das gesamte Benutzerinterface gestaltet ist. Die OpenGL-Renderfläche wurde direkt in ein Fensterobjekt eingebettet. Erstere wird dann mit 3D-Objekten gefüllt, welche aus den in die *CnfFile*-Instanz geladenen Informationen erstellt werden. Zur Laufzeit ist es möglich, verschiedene Systeme darzustellen. Zu diesem Zweck werden die in *CnfFile* befindlichen Werte überschrieben und erneut an die Renderfläche gesendet.

4.3.2 Benutzerspezifische Anpassung

Es wird bei einigen Features sicherlich notwendig sein, sie für verschiedene Benutzer anzupassen. Selbstverständlich werden verschiedene Simulationsprogramme verschieden strukturierte Ausgabedateien produzieren. Die Routinen zum Einlesen der Molekülinformationen sind jedoch allesamt in einer einzigen Funktion der Klasse *CnfFile*, namentlich

loadCnfFile(), untergebracht und können bei Bedarf angepasst werden. In einer neueren Version wurde auch ein Mechanismus integriert, um QMGA mit beliebigen, zur Laufzeit austauschbaren, Einleseroutinen auszustatten. Es ist somit sehr einfach, QMGA an die Bedürfnisse eines jeden Simulationsprogramms anzupassen, und demnach können im Prinzip nahezu beliebige Dateiformate unterstützt werden. Einzig Informationen zur Position und zur Ausrichtung für jeden Körper sind notwendig. Sollten die darzustellenden Körper über keine Ausrichtungsinformationen verfügen, zum Beispiel wenn kugelförmige Objekte behandelt werden, so können diese durch beliebige Werte vorgegeben werden, ohne dass dieses Einfluss auf die Funktion von QMGA hätte.

Zum jetzigen Zeitpunkt sind fünf Dateiformate (gbmega, gbmegaBiax, lammps1, lammps2 und cinacchi) in QMGA fest implementiert, wobei der weiter oben erwähnte Mechanismus genutzt wurde. Als Beispiel werden im Folgenden gbmega und gbmegaBiax etwas näher vorgestellt:

gbmega

Hierbei handelt es sich um das Ausgabeformat des unter anderem von W. Smith, M. P. Allen, M. Warren, M. Wilson, G. Germano et al. entwickelten Molekularsimulationsprogramms GBmega.[92]

- Kopf
  int (Anzahl der Moleküle in der Datei)
  double (x-Länge der Simulationsbox)
  double (y-Länge der Simulationsbox)
  double (z-Länge der Simulationsbox)
  2 doubles (für „moving boundary conditions“)

- Molekülinformationen
  12 doubles ($\vec{r}_i$, $\vec{v}_i$, $\vec{e}_i$, $\vec{u}_i$), int (Zähler), int (Etikett als Typbezeichner, optional)

Dabei ist $\vec{r}_i$ die Position des Moleküls i, $\vec{v}_i$ seine Geschwindigkeit, $\vec{e}_i$ seine Orientierung und $\vec{u}_i$ seine Orientierungsgeschwindigkeit. Von allen diesen Zahlen werden zur Zeit die Längen der Simulationsbox, $\vec{r}_i$, $\vec{e}_i$ und ein evtl. vorhandener Typbezeichner für die Visualisierung genutzt.

Ein Beispiel: Abb. 4.12 gibt die Textdatei wieder, welche das Testsystem erzeugt, das eingangs in Abb. 4.2 auf dem Bildschirmfoto zu sehen ist.

gbmegaBiax

Dieses Format ist nahezu identisch zum vorgenannten, lediglich die Orientierungsinformationen $\vec{e}_i$ werden nicht durch einen dreidimensionalen Vektor angegeben. An diese Stelle tritt ein Quaternion $\vec{q}_i$ (spezieller vierdimensionaler Vektor), um auch Körper darstellen

```
17
9.0
9.0
9.0
0.000000000000     0.00000000
-4.0    0.0    0.0       0 0 0       0.0    1.0    0.0       0 0 0 1
-3.0    0.0    0.0       0 0 0       0.0    2.0    1.0       0 0 0 2
-2.0    0.0    0.0       0 0 0       0.0    1.0    1.0       0 0 0 3
-1.0    0.0    0.0       0 0 0       0.0    1.0    2.0       0 0 0 4
 0.0    0.0    0.0       0 0 0       0.0    0.0    1.0       0 0 0 5
 1.0    0.0    0.0       0 0 0       0.0   -1.0    2.0       0 0 0 6
 2.0    0.0    0.0       0 0 0       0.0   -1.0    1.0       0 0 0 7
 3.0    0.0    0.0       0 0 0       0.0   -2.0    1.0       0 0 0 8
 4.0    0.0    0.0       0 0 0       0.0   -1.0    0.0       0 0 0 9
 0.0   -4.0    0.0       0 0 0       1.0    0.0    0.0       0 0 0 10
 0.0   -3.0    0.0       0 0 0       2.0    0.0    1.0       0 0 0 11
 0.0   -2.0    0.0       0 0 0       1.0    0.0    1.0       0 0 0 12
 0.0   -1.0    0.0       0 0 0       1.0    0.0    2.0       0 0 0 13
 0.0    1.0    0.0       0 0 0      -1.0    0.0    2.0       0 0 0 14
 0.0    2.0    0.0       0 0 0      -1.0    0.0    1.0       0 0 0 15
 0.0    3.0    0.0       0 0 0      -2.0    0.0    1.0       0 0 0 16
 0.0    4.0    0.0       0 0 0      -1.0    0.0    0.0       0 0 0 17
```

Abbildung 4.12: An dieser Stelle ist die Textdatei abgedruckt, welche das in Abb. 4.2 gezeigte Testsystem erzeugt. Das Dateiformat entspricht der Ausgabe des Molekularsimulationsprogramms GBmega.

zu können, die nicht über eine ausgezeichnete Rotationsachse verfügen, im Wesentlichen also biaxiale Ellipsoide.

- Kopf
  int (Anzahl der Moleküle in der Datei)
  double (x-Länge der Simulationsbox)
  double (y-Länge der Simulationsbox)
  double (z-Länge der Simulationsbox)
  2 doubles (für „moving boundary conditions“)

- Molekülinformationen
  12 doubles ($\vec{r}_i$, $\vec{v}_i$, $\vec{q}_i$, $\vec{u}_i$), int (Zähler), int (Etikett als Typbezeichner, optional)

Eine weitere generell für alle Formate geltende Bemerkung betrifft die Angabe der Simulationsboxgrenzen. Neben den üblicherweise quaderförmigen Simulationsboxen sind auch solche in Form eines Parallelepipeds in Gebrauch. Um dieser Tatsache Rechnung zu tragen und den Benutzern die Darstellung einer schiefwinkligen Simulationsbox zu ermöglichen, werden die Boxgrenzen generell in einer 3×3 Matrix gespeichert. Diese entspricht drei Dreiervektoren, welche einen Parallelepiped aufspannen. Im Falle einer Diagonalmatrix ergibt sich somit eine rechtwinklige, quaderförmige Box.

Die Einleseroutine für die gbmega und gbmegaBiax Formate unterscheidet die Angabe von je einem double Wert pro Zeile für die Boxparameter im Dateikopf oder dreien. Ist nur

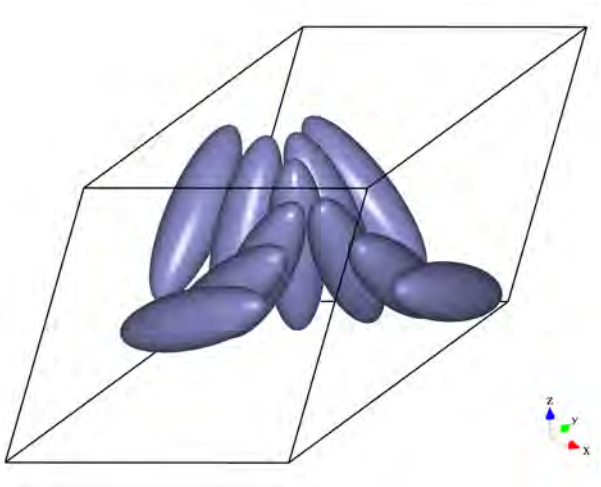


Abbildung 4.13: Willkürlich gewähltes Testsystem zur Veranschaulichung schiefsymmetrischer Simulationsboxen. Die Matrix, welche in der gezeigten Box resultiert, lautet $\hat{B}_{\text{box}} = \begin{pmatrix} 5.0 & 1.0 & 1.0 \\ 1.0 & 5.0 & 1.0 \\ 1.0 & 1.0 & 5.0 \end{pmatrix}$. Die dargestellten Ellipsoide dienen der Wahrung des richtigen räumlichen Eindrucks und sind ansonsten ohne jede Bedeutung.

jeweils ein Wert angegeben, wird dieser auf die Diagonale der 3×3 Matrix eingetragen. Im Fall von dreien wird die gesamte Matrix entsprechend gefüllt. In diesem Sinne erzeugen die folgenden beiden Dateiköpfe die gleiche Darstellung der Simulationsbox (ein Würfel mit Kantenlänge drei):

- Kopf
  10
  3.0 0.0 0.0
  0.0 3.0 0.0
  0.0 0.0 3.0
  0.0 0.0

bzw.

- Kopf
  10
  3.0
  3.0
  3.0
  0.0 0.0

Abb. 4.13 zeigt ein Testsystem zur Veranschaulichung einer schiefwinkligen Simulationsbox.

Ein weiteres Feature, welches wahrscheinlich an verschiedene Benutzer angepasst werden muss, ist die Farbkodierung der Moleküle. Auch an dieser Stelle kann behauptet wer-

den, dass eine solche Anpassung einfach durchzuführen ist. Der gesamte Programmcode, der für die Färbung zuständig ist, befindet sich derzeit in drei recht kurzen Funktionen. Eine dieser Funktionen ist Teil der Klasse CnfFile und trägt den Namen *colorizeMolecules()*. Ihre einzige Aufgabe ist es, nacheinander alle vorhandenen Molekülinstanzen an eine Funktion der Klasse Colormap zu schicken. Diese Funktion, *setColor()*, ist es auch, innerhalb derer am ehesten Anpassungen vorgenommen werden müssen. An dieser Stelle soll auch darauf hingewiesen werden, dass *setColor()* in zwei überladenen Versionen implementiert wurde, um sowohl die Kodierung nach Typ als auch die Kodierung nach Orientierung zu ermöglichen. Abb. 4.14 zeigt die derzeitige Version von *setColor()*, welche die zuvor beschriebene Farbkodierung in Abhängigkeit von der Orientierung des Moleküls realisiert.

4.4 Performance

4.4.1 Szenengraph gegen direktes Rendern

Zu Beginn des Projekts wurde die Renderfläche von QMGA in einfacher Weise unter Zuhilfenahme von SiMs Coin3D-Toolkit realisiert.[93] Coin3D ist eine Quellcode-offene Bibliothek bestehend aus einer Sammlung von Objekten wie Lichtquellen, einiger Standardkörper (Kegel, Kugel, usw.), Materialien und einem Mechanismus, um diese in einem Szenengraph zu vereinen und darzustellen. Coin3D ist in Verhalten und Benutzung sehr ähnlich zu SGI's OpenInventor,[94] welcher zu Beginn des Projekts noch kommerziell vertrieben wurde. Ebenfalls von Coin3D geregelt wird auch der Einbau der Renderfläche in den Qt-Kontext. Die angezeigte Szene konnte mit Hilfe der Coin3D-Funktionalität komplett erstellt und angezeigt werden. Die hauptsächliche Arbeit wurde hierbei in einer einzigen for-Schleife über alle Molekülobjekte des Systems erledigt.

Ein weiteres Beispiel für die Hilfe, welche vordefinierte Bibliotheken darstellen, ist der Mechanismus zum Speichern eines Bildschirmfotos der aktuellen Ansicht im Renderbereich. Qt stellt einen betriebssystemnativen Dateidialog und Coin3D die nötigen Funktionen, um den Inhalt des Renderbereichs in einer großen Zahl von Bildformaten zu exportieren. Insgesamt kann dann mit wenigen Zeilen Programmcode ein sehr komplexer Vorgang in einfacher Weise ausgeführt werden.

Obwohl dieser einfache Ansatz, welcher auch von anderen Molekulargraphikprogrammen angewendet wird,[52] für Systemgrößen bis etwa 10^4 Teilchen gut funktioniert, erhöht sich die Zeit, die das Rendern eines einzigen Bildes in Anspruch nimmt, mit größeren Systemen stark. Je nach Rechnerkonfiguration wird die Visualisierung von Systemen ab etwa 10^5 Teilchen zu langsam um das Programm sinnvoll einzusetzen. Dabei bleibt zu bemerken, dass QMGA auch unter solch extremen Bedingungen noch stabil arbeitete,

```
Molecule* mga::Colormap::setColor( Molecule* moleculeTmp,
                                   vector<double> &director,
                                   vector<vector<float> > *models ) const
{
    if( (moleculeTmp != 0) && (director.size() == 3) )
    {
        double orientationX = moleculeTmp -> getOrientationX();
        double orientationY = moleculeTmp -> getOrientationY();
        double orientationZ = moleculeTmp -> getOrientationZ();

        int numOfMapLines = redVector.size();
        float scalarProductOD = fabs( orientationX*director.at(0) +
                                      orientationY*director.at(1) +
                                      orientationZ*director.at(2)   );
        if( scalarProductOD > 1 ) { scalarProductOD = 1; }
        int mapLineNr = int(acos(scalarProductOD)/M_PI*2*(numOfMapLines));
        if( mapLineNr == 90 ) { mapLineNr = 89; }
        moleculeTmp -> setRGB( getRed( mapLineNr ),
                               getGreen( mapLineNr ),
                               getBlue( mapLineNr ) );
        moleculeTmp -> setColorIndex( mapLineNr );
        if( models != 0 )
        {
            int typeTmp = moleculeTmp->getType();
            if( typeTmp >= 0 && typeTmp < int(models->size()) )
            {
                if( models->at(typeTmp).at(11) != 0.0 )
                {
                    setColor( moleculeTmp, models );
                }
            }
        }
    }
    else { cerr << "setColor: no molecule, or director corrupt" << endl; }
    return( moleculeTmp );
}
```

Abbildung 4.14: Die Funktion *setColor()* der Klasse *Colormap*. Die Variablen *orientationX/Y/Z* sind die Komponenten des Vektors, der die Orientierung des zu färbenden Objekts im Raum angibt. Das Vektorobjekt namens *director* wurde in einem vergangenen Arbeitsschritt berechnet und repräsentiert den gemittelten Direktor der Mesophase. Die Farbpalettenklasse *Colormap* selbst beinhaltet eine Liste von RGB-kodierten Farbwerten, welche zeilenweise angeordnet sind. Die Zeilennummer der Farbe, welche das aktuelle Objekt zugewiesen bekommen soll, entspricht dem kleineren Winkel zwischen Orientierungsvektor und Direktor und muss erst berechnet werden. Dies geschieht, wie im Programmcode zu lesen, über das Skalarprodukt der zwei Vektoren.

lediglich die Framerate[†] erheblich zu niedrig war. Aus diesem Grund wurde die Renderfläche neu gestaltet und von Grund auf neu geschrieben, wobei auf die Bibliothek Coin3D vollständig verzichtet wurde. Des weiteren wurde der hauptsächliche Anwendungsbereich, nämlich die Darstellung vieler gleicher und dicht gepackter Objekte, berücksichtigt. Alle bisher beschriebenen Features wurden beibehalten oder sogar erst in der zweiten Version eingebracht. Eine Ausnahme in diesem Punkt stellt die Auswahl an Bildformaten dar, welche für Bildschirmfotos zur Verfügung steht. Diese wurde auf die Formate portable network graphics (PNG) und PostScript begrenzt, da es den Aufwand nicht gerechtfertigt hätte, alle von Coin3D zur Verfügung gestellten Formate (JPEG, TIFF, BMP, etc.) ohne diese Bibliothek zu implementieren.

Für die Gestaltung der Renderfläche ist es unter anderem wichtig, Ort und Blickrichtung des Betrachters im dreidimensionalen Raum angeben zu können, damit es möglich ist das, betrachtete System zu drehen. Da bei Coin3D dieser Teil schon eingebaut war, musste er nun neu entwickelt werden. Für die Spezifikation dieser Informationen gibt es mehrere Möglichkeiten. Eine davon ist die Angabe dreier Eulerwinkel α, β, γ und diese erschien als die praktikabelste insbesondere auch im Hinblick auf die Benutzeroberfläche. Eulerwinkel beschreiben drei sukzessive Drehungen um drei Koordinatenachsen und somit den Übergang des Ausgangssystems in das gedrehte System, welches auf dem Bildschirm angezeigt wird.[95]

Quaternionen sind von der technischen Seite sicherlich die beste Lösung, die Angabe eines Quaternions in der Benutzeroberfläche, um eine bestimmte Ansicht zu erzielen, gestaltet sich hingegen für den Benutzer als eher mühselig und wenig intuitiv. Es sei an dieser Stelle jedoch darauf hingewiesen, dass Quaternionen durchaus ihr Platz in QMGA zukommt, nämlich bei der Ausrichtung der einzelnen Objekte innerhalb eines Systems; siehe Abs. 4.3.2.

Bei der Gestaltung der Renderfläche wurden einige Ansätze verfolgt, um die Rechenlast zu reduzieren. Im Folgenden werden die wichtigsten vorgestellt und auch einige Benchmarks präsentiert.

4.4.2 Display Lists gegen Vertex Buffer Objects

Um die Rendergeschwindigkeit für die einzelnen Objekte zu verbessern, wurden zu deren Beschreibung sowohl Display Lists (DL) als auch Vertex Buffer Objects (VBO) getestet. Eine OpenGL-DL bereitet ein Modell im Grafikkartenspeicher vor, so dass es mit einem einzigen OpenGL-Aufruf mit seinem Index gezeichnet werden kann. Vor dem eigentlichen Zeichenvorgang wird die Position und die Ausrichtung festgelegt, um das Objekt in gewünschter Weise zu rendern. Demzufolge ist es sehr einfach, das Modell auszutauschen,

[†]Anzahl der Bilder, die pro Sekunde gerendert werden. Die Angabe erfolgt üblicherweise in fps (frames per second).

da der restliche Rendercode identisch bleiben kann. Es muss nur eine weitere DL erstellt und in die Grafikkarte geladen werden.

OpenGL-VBOs sind weitgehend ähnlich in der Benutzung, jedoch sind VBOs etwas weniger beladen als DLs: Sie beinhalten weniger Informationen über das Objekt, beispielsweise keine Transformationen und Materialien. Damit kann der Grafikkartentreiber die notwendige Arbeit zum Testen, ob diese Informationen überhaupt vorhanden sind und ob sie beachtet werden müssen, vermeiden. Der erreichbare Geschwindigkeitsgewinn bei einem Wechsel von DLs nach VBOs hängt stark von dem betreffenden Programm und ebenfalls von dem Computersystem ab. Auf dem benutzten Testsystem zeigten die VBOs eine durchweg, wenn auch nur geringfügig, bessere Leistung; zum Vergleich siehe Tab. 4.1 und Abb. 4.16. VBOs bringen zusätzlich noch den Vorteil, dass der Programmcode etwas kürzer und klarer formuliert werden kann. DLs hingegen wurden aus dem aktuellen OpenGL-Standart gestrichen und sind offiziell nicht weiter unterstützt. Ein Wechsel hin zu VBOs scheint somit zumindest mittelfristig notwendig zu werden.

4.4.3 Detailstufe

Eine gängige Methode zur Optimierung ist die Einstellung der Detailstufe. Die üblicherweise verwendete Bezeichnung für diesen Vorgang ist „level of detail“ oder auch kurz LOD. Hierbei werden Objekte nahe des Betrachters mit größerer Detailstufe gerendert als weiter entfernte. Der Sinn der Methode ist, dass letztere ohnehin kleiner und somit detailärmer erscheinen. Wir verwenden nicht einen linearen Ansatz für das LOD, sondern einen sich selbst anpassenden. Der Benutzer wählt dabei eine der Qualitätsstufen, wie sie in Abs. 4.2.10 gezeigt wurden. Nun wird allerdings zusätzlich noch eine maximale und minimale Detailstufe definiert, welche nicht notwendigerweise den 5 erwähnten Stufen entspricht. Fällt im weiteren Programmablauf die Framerate unter einen Schwellenwert, so wird die Detailstufe, beginnend bei den entferntesten Objekten, sukzessive erniedrigt, bis entweder die Rate die Schwelle wieder überschreitet, oder alle Teilchen in der minimalen Detailstufe angezeigt werden. Andererseits wird aber auch bei Überschreiten einer Schwelle die Detailstufe wieder heraufgesetzt, wobei nun bei den am nächsten gelegenen Objekten begonnen wird. Da LOD, gerade zum Beispiel bei großen Ausdrucken, unerwünschte Grafikverschlechterungen hervorrufen kann, sollte dieses Feature vom Benutzer deaktiviert werden, da dies bei einem Bildschirmfoto nicht automatisch geschieht.

Die Anwendung von LOD ist auch im Falle von QMGA sinnvoll, obwohl wir für die Darstellung eine orthographische Ansicht (parallel, ohne Fluchtpunkt) gewählt haben. Entferntere Objekte erscheinen zwar nicht kleiner, aber wegen der normalerweise hohen Teilchendichte ist die Wahrscheinlichkeit, dass entferntere Objekte von näheren ganz oder teilweise verdeckt werden, recht hoch.

4.4.4 Occlusion query

Wie schon in Abs. 4.4.3 über LOD erwähnt, werden in dichten Systemen die weiter vom Betrachter entfernten Objekte von den näher liegenden möglicherweise verdeckt. Je dichter und größer ein System, je größer der Anteil der Objekte, die vollständig verdeckt werden und dementsprechend nicht gerendert werden müssten. Diesen Umstand macht sich die hier beschriebene Optimierungsmethode zu Nutze. Die benötigte Funktionalität wird von der OpenGL-Erweiterung „GL_ARB_occlusion_query“ bereitgestellt. Sie fragt die Grafikkarte, ob die nächste Anweisung gerendert werden muss oder nicht. Da es nicht sinnvoll ist, für jedes einzelne Objekt diese Anfrage zu starten, wird der gesamte mit zu rendernden Teilchen gefüllte Raum in $n \times n \times n$ Zellen aufgeteilt. Alle Teilchen werden nun entsprechend ihrer Position diesen Zellen zugeteilt. Die Anfrage an die Grafikkarte wird im Weiteren jeweils für eine gesamte Zelle ausgeführt. Diese wird bei einer negativen Antwort komplett verworfen, ansonsten komplett gezeichnet. Der gesamte Rendervorgang wird zellenweise und von vorne nach hinten ausgeführt. Auf diese Weise ist es möglich, auf das Zeichnen vieler Boxen und damit vieler Objekte zu verzichten. Selbstverständlich kostet es ebenfalls Rechenzeit, die Moleküle zu sortieren und die Anfragen an die Grafikkarte abzuarbeiten. In Abs. 4.4.6 wird gezeigt, dass sich diese Optimierung tatsächlich erst ab einer gewissen Systemgröße auszahlt.

4.4.5 Backface culling

Es ist nicht notwendig, die Rückseite undurchsichtiger Körper zu zeichnen. Die OpenGL-Bibliothek ist in der Lage, dies mit einem einfachen Aufruf zu erledigen und damit in unserem Fall den Renderaufwand um 50% zu reduzieren. Der tatsächliche Performancegewinn beträgt jedoch lediglich etwa 10%, da die benötigten internen Berechnungen zum Herausfinden der Rückseite eines Objekts in der aktuellen Ansicht nahezu so rechenintensiv sind wie der vermiedene Rendervorgang selbst.

4.4.6 Benchmarkergebnisse

Die in diesem Abschnitt präsentierten Testergebnisse wurden auf folgendem Computersystem durchgeführt:

- Prozessor: AMD Athlon 64 3500+ mit 2.2GHz
- Arbeitsspeicher: 1GB PC3200 SD-RAM
- Grafikkarte: NVidia 6600GT mit 128MB DDR3 RAM

Um aussagekräftige und stabile Messergebnisse generieren zu können wurde eine Benchmarkfunktionalität in QMGA implementiert. Wird diese aktiviert, läuft eine Serie von

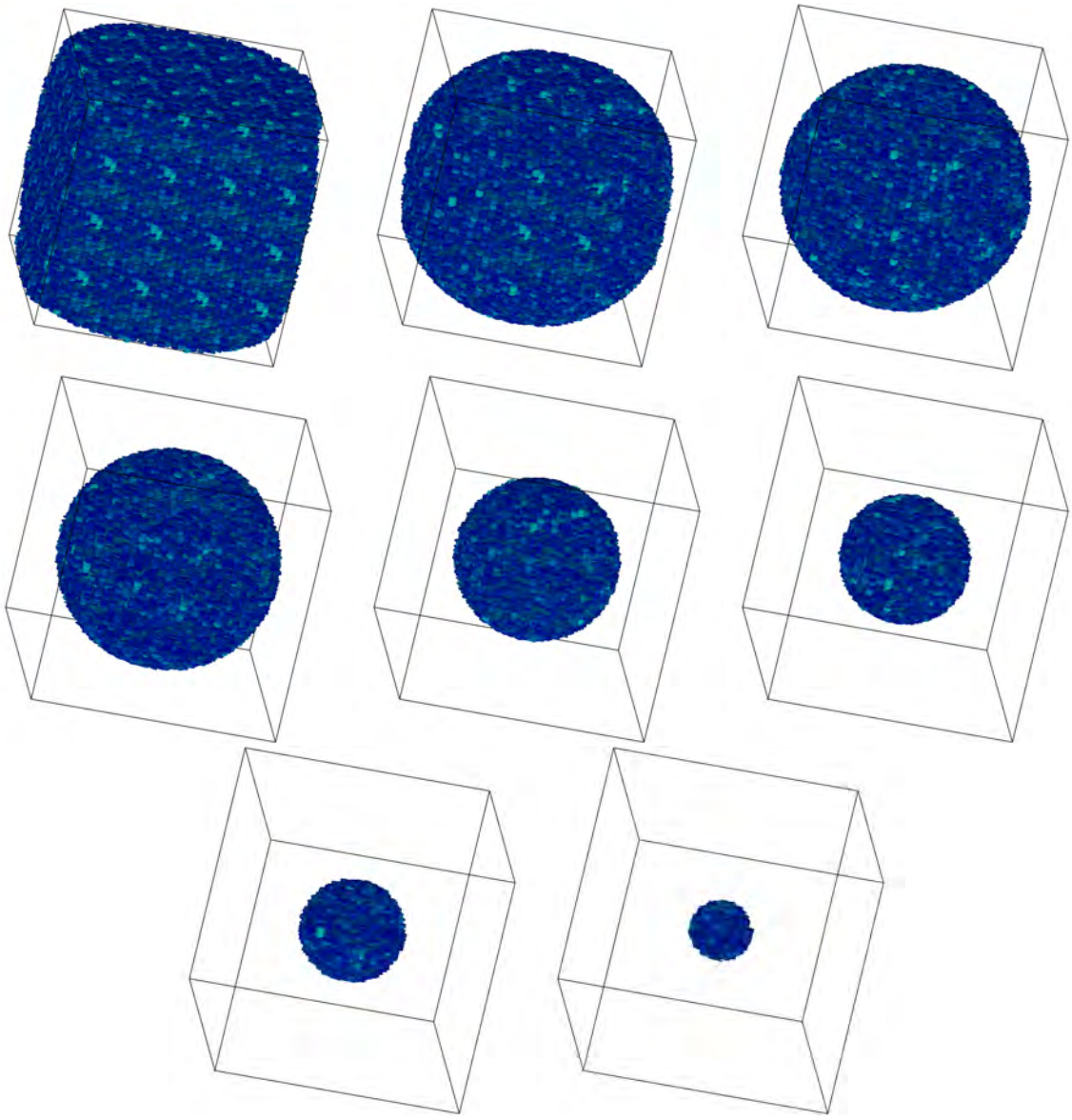


Abbildung 4.15: Die Testsysteme, die zur Benchmarkmessung von QMGA verwendet wurden. Von links nach rechts und oben nach unten sinkt die Systemgröße von 140 000 Molekülen auf 1 000 Moleküle. Das System wurde aus einer kleineren realen kolumnaren Phase durch Replikation erzeugt.

zufälligen Drehungen ab. Die Anzahl der Drehungen ist einstellbar. Während des Benchmarks wird die jeweils aktuelle Rendergeschwindigkeit gemessen und deren Mittelwert sowie Standardabweichung berechnet.

Das Testsystem bestand aus 140 000 diskotischen Molekülen in einer kolumnaren Phase. Um die Abhängigkeit der Rendergeschwindigkeit von der Systemgröße zu ermitteln, wurde das Testsystem sukzessive bis auf 1 000 Moleküle verkleinert. Die Moleküle wurden dabei nach ihrem Abstand zum Mittelpunkt des Systems sortiert. Die resultierenden Systeme nehmen wegen dieser Sortierung eine kugelförmige Gestalt an; siehe Abb. 4.15.

$N/1000$	DL	VBO	DL+OC	VBO+OC
1	485,2	585,7	75,0	79,6
5	95,6	108,7	49,9	50,3
10	48,8	54,8	37,4	38,0
20	24,7	27,6	26,8	27,4
50	9,9	11,1	16,3	17,4
70	7,1	8,0	14,0	14,2
100	4,9	5,4	11,5	11,8
140	3,5	3,6	9,1	9,3

Tabelle 4.1: Benchmark-Ergebnisse für ein System N diskotischer Ellipsoide in kolumnarer Phase. Es wurden sowohl Display Lists (DL) als auch Vertex Buffer Objects (VBO) jeweils mit und ohne Occlusion Culling (OC) getestet. Alle Fehler liegen unter 5%.

Für jedes System wurden zwei Messreihen zu je 12 mal 500 zufälligen Drehungen bei einer mittleren Qualitätsstufe durchgeführt, je System eine Messreihe mit und eine ohne aktiviertes Occlusion Culling (OC). Die sich aus der Standardabweichung ergebenden Fehler lagen bei allen Messreihen unter 5% und sind aus diesem Grund in Abb. 4.16 nicht zu erkennen. Die Resultate zeigen, dass die Aktivierung des OC die Renderperformance für Systeme ab ca. 10 000 Teilchen deutlich steigert.

Die Messwerte sind in Tab. 4.1 und Abb. 4.16 zusammengefasst. Es ist zu erkennen, dass QMGA sich auf dem verwendeten Computersystem auch bei mehr als 100 000 Teilchen noch recht gut verhält. Zum Vergleich: Eine Rendergeschwindigkeit ab ca. 8 fps fühlt sich in der Benutzung praktisch flüssig, wenn auch etwas viskos, an. Ab etwa 5 fps und weniger wird das System merklich bis störend träge und beginnt bei Drehungen deutlich zu ruckeln.

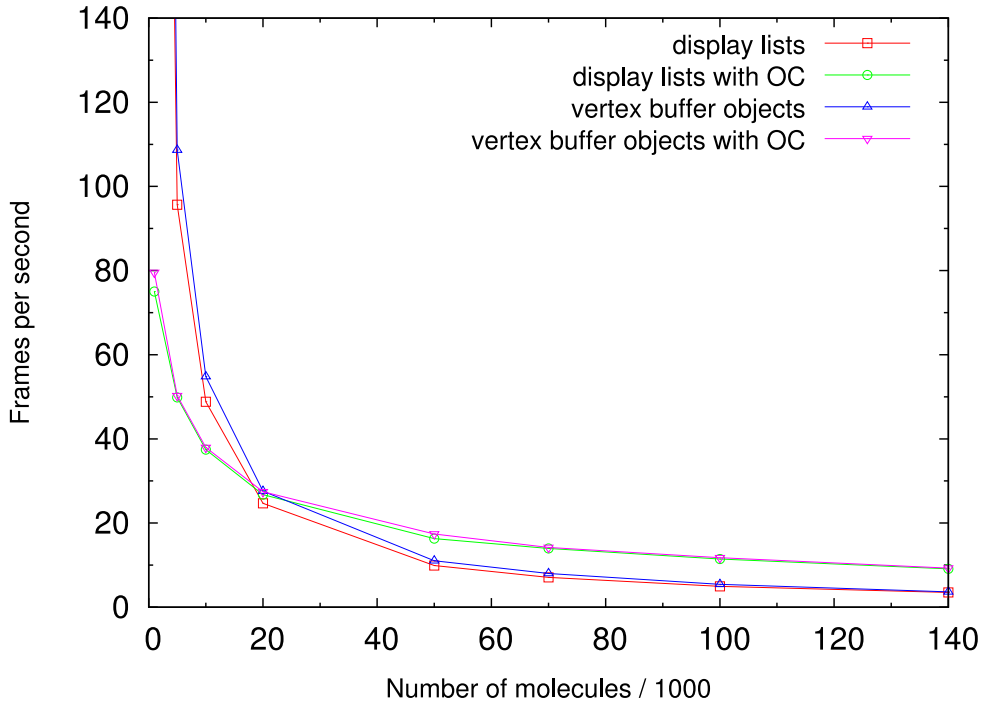


Abbildung 4.16: Graphische Darstellung der Benchmarkergebnisse aus Tab. 4.1. Man beachte die Überschneidung der Graphen bei ca. 20 000 Teilchen. Die Fehlerbalken wurden ausgelassen, da sie mit Ausnahme der Datenpunkte für 1 000 Teilchen kleiner als die Datenpunktsymbole sind.

Ausblick

Die Berechnungen zum Lorentzgas aus Kapitel 2 lassen nur den Schluss zu, dass in den betrachteten Prozessen keine Asymmetrien bei großen Fluktuationen auftauchen. Die zugrundeliegende Theorie scheint auch schon die Antwort zu diesem Ergebnis zu geben, da die einzelnen simulierten Elektronen nicht miteinander wechselwirken. Tatsächlich haben mittlerweile weitere Computerexperimente an anderen, wechselwirkenden, Systemen gezeigt, dass dort Asymmetrien festzustellen sind.[29,96–99] Man könnte weitere solcher Systeme untersuchen. Zum Beispiel die Berechnung des Wärmeflusses durch eine Oszillatorkette oder durch ein Becken mit einer Lennard-Jones-Flüssigkeit, wenn die gegenüberliegenden Enden auf unterschiedlicher Temperatur gehalten werden, oder auch den Fluss kinetischer Energie in einer senkrechten periodischen Röhre im Schwerefeld, wenn diese zum Teil mit harten Kugeln und festen Hindernissen zur Durchmischung gefüllt wird.

Den Ergebnissen zu den Systemen harter Kugeln aus Kapitel 3 ist nur wenig hinzuzufügen. Einige Punkte sind dennoch auch an dieser Stelle offen. Insbesondere blieben bisher die beobachteten kleinen Abweichungen in den simulierten Verteilungen ungeklärt. Ausführlicher wird auf diese Thematik in der Diskussion in Abs. 3.7 eingegangen. Eine mögliche Methode, um diese Anomalien zu erforschen, wäre, die Systeme mit anderen Randbedingungen zu simulieren. Eine Änderung der Topologie von einer quadratischen zweidimensionalen Box mit periodischen Randbedingungen hin zu einer Simulationszelle, welche auf der Oberfläche einer dreidimensionalen Kugel liegt, wäre eine Möglichkeit. Auch eine kubische Simulationsbox mit periodischen Randbedingungen lässt sich in dieser Weise auf die Oberfläche einer vierdimensionalen Kugel abbilden. Die Entwicklung der Theorie ist aktuell noch nicht vollständig abgeschlossen, aber eine endgültige Formulierung wird in naher Zukunft erwartet.

Die Software QMGA erfreut sich zunehmender Beliebtheit in der Gemeinschaft der mit *coarse graining* simulierenden Arbeitsgruppen, so dass die Entwicklung als notwendig und nützlich anzusehen ist.[86,100–107] Dies zeigt ebenfalls die Entwicklung eines weiteren Visualisierungsprogramms namens BioVEC, welches an dieser Stelle nicht unerwähnt bleiben soll. Im Gegensatz zu QMGA zielt BioVec allerdings primär auf eine *coarse-grained* Darstellung von Biomakromolekülen.[108] Weiteres Entwicklungspotential besteht insbesondere bei den verwendeten Programmierumgebungen und Techniken. Zu Beginn der Entwicklung war Qt in der dritten Generation noch der Standard, mittlerweile ist Qt3

jedoch praktisch vollständig durch seinen Nachfolger Qt4 abgelöst worden. Auch wenn die alten Bibliotheken weiterhin verfügbar sind, scheint es dennoch sinnvoll, eine Portierung hin zur neuen Umgebung durchzuführen. Ein weiterer und wesentlich dringlicherer Punkt wäre, die Verwendung der Display Lists (DL) durch Vertex Buffer Objects (VBO) zu ersetzen. Erstere sind nicht mehr Bestandteil des aktuellen OpenGL-Standards. Es ist zwar nicht zu erwarten, dass die Unterstützung in den Graphikkartentreibern in naher Zukunft eingestellt wird, aber ein Wechsel scheint an dieser Stelle dennoch eine gute Idee. Einige grundsätzliche Verbesserungen der Benutzerfreundlichkeit wären ebenso angebracht. Zur Zeit kann QMGA nur auf Linux-basierten Systemen kompiliert werden. Unter Windows kann QMGA mit Hilfe von Cygwin,[109] einer Linux-ähnlichen Umgebung für Windows, kompiliert und genutzt werden. Eine native Version für Windows oder MacOS X ist derzeit nicht sinnvoll möglich. Um weitere Betriebssysteme nativ unterstützen zu können, wäre ein Wechsel zu Qt4 grundlegende Voraussetzung, da Qt3, im Gegensatz zu Qt4, nur unter Linux frei für nicht kommerzielle Zwecke zur Verfügung steht. Ebenfalls wünschenswert wäre in diesem Zusammenhang die Erstellung einer in binärer Form verteilten Version von QMGA, um weniger versierten Benutzern die Möglichkeit zu geben, QMGA ohne ein notwendiges Kompilieren zu Installieren. Sinnvoll wäre demnach ein Windows-Installationspaket und evtl. der Eingang von QMGA in ein Repositorium einer einschlägigen Linuxdistribution. Einige der genannten Punkte sollen im Rahmen einer Postdoc-Anstellung umgesetzt werden. Insbesondere der Portierung von Qt3 nach Qt4 und dem Wechsel von DL nach VBO wird die Priorität gegeben.

Literaturverzeichnis

[1] M. P. Allen, D. J. Tildesley, Computer Simulation of Liquids, Paperback Edition, Oxford University Press, Oxford, 1989.

[2] A. Rahman, Correlations in the motion of atoms in liquid argon, Phys. Rev. 136 (1964) A405–A411.

[3] B. J. Alder, T. E. Wainwright, Phase transition for a hard sphere system, J. Chem. Phys. 27 (1957) 1208–1209.

[4] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (1953) 1087–1092.

[5] M. N. Rosenbluth, A. W. Rosenbluth, Further results on Monte Carlo equations of state, J. Chem. Phys. 22 (1954) 881–884.

[6] H. C. Andersen, Molecular dynamics simulations at constant pressure and/or temperature, J. Chem. Phys. 72 (1980) 2384–2393.

[7] S. Nosé, A unified formulation of the constant temperature molecular dynamics methods, J. Chem. Phys. 81 (1984) 511–519.

[8] W. G. Hoover, Canonical dynamics: Equilibrium phase-space distributions, Phys. Rev. A. 31 (1985) 1695–1697.

[9] A. Sergi, M. Ferrario, D. Costa, Reversible integrators for basic extended system molecular dynamics, Mol. Phys. 97 (1999) 825–832.

[10] D. Frenkel, B. Smit, Understanding Molecular Simulation – From Algorithms to Applications, 2nd Edition, Academic Press, San Diego, 2002.

[11] W. C. Swope, H. C. Andersen, P. H. Berens, Computer-simulation method for the calculation of equilibrium-constants for the formation of physical clusters of molecules — application to small water clusters, J. Chem. Phys. 76 (1982) 637–649.

[12] M. E. Tuckerman, G. J. Martyna, B. J. Berne, Reversible multiple time scale molecular dynamics, J. Chem. Phys. 97 (1992) 1990–2001.

[13] H. F. Trotter, On the product of semi–groups of operators, Proc. Am. Math Soc. 10 (1959) 545.

[14] D. C. Rapaport, The Art of Molecular Dynamics Simulation, 2nd Edition, Cambridge University Press, Cambridge, 2004.

[15] M. Marin, P. Cordero, An empirical assessment of priority-queues in event driven molecular dynamics simulation, Comput. Phys. Commun. 92 (1995) 214–224.

[16] G. Paul, A complexity O(1) priority queue for event driven molecular dynamics simulations, J. Comput. Phys. 221 (2007) 615–625.

[17] N. M. Josuttis, The C++ Standard Library — A Tutorial and Reference, Addison-Wesley, Boston, 1999.

[18] B. D. Lubachevsky, How to simulate billiards and similar systems, J. Chem. Phys. 94 (1991) 255–283.

[19] M. Marin, D. Risso, P. Cordero, Efficient algorithms for many-body hard particle molecular dynamics, J. Comput. Phys. 109 (1993) 306–317.

[20] M. Isobe, Simple and efficient algorithm for large scale molecular dynamics simulation in hard disk system, Int. J. Mod. Phys. C 10 (1999) 1281–1293.

[21] S. Miller, S. Luding, Event-driven molecular dynamics in parallel, J. Comput. Phys. 193 (2003) 306–315.

[22] D. G. Luchinsky, P. V. E. McClintock, Irreversibility of classical fluctuations studied in analogue electrical circuits, Nature 389 (1997) 463–466.

[23] L. Onsager, S. Machlup, Fluctuations and irreversible processes, Phys. Rev. 91 (1953) 1505–1512.

[24] L. Bertini, A. De Sole, D. Gabrielli, G. Jona-Lasinio, C. Landim, Fluctuations in stationary nonequilibrium states of irreversible processes, Phys. Rev. Lett. 87 (2001) 040601.

[25] L. Bertini, A. De Sole, D. Gabrielli, G. Jona-Lasinio, C. Landim, Macroscopic fluctuation theory for stationary non-equilibrium states, J. Stat. Phys. 107 (2002) 635–675.

[26] A. Gamba, L. Rondoni, Current fluctuations in the nonequilibrium Lorentz gas, Physica A 340 (2004) 274–282.

[27] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, Version 2.2, High Performance Computing Center Stuttgart (HLRS), Stuttgart, 2009, http://www.mpi-forum.org (accessed Oct 13, 2010).

[28] D. Brown, J. H. R. Clarke, A comparison of constant energy, constant temperature and constant pressure ensembles in molecular dynamics simulations of atomic liquids, Mol. Phys. 51 (1984) 1243–1252.

[29] C. Paneni, D. J. Luding, L. Rondoni, Temporal asymmetry of fluctuations in nonequilibrium steady states, J. Chem. Phys. 124 (2006) 114109.

[30] J. C. Maxwell, Illustrations of the dynamical theory of gases. Part 1. On the motions and collisions of perfectly elastic spheres, Phil. Mag. 19 (1860) 19–32.

[31] P. W. Atkins, Physical Chemistry, Oxford University Press, Oxford, 1978.

[32] J. C. Maxwell, On the dynamical theory of gases, Phil. Trans. R. Soc. London 157 (1860) 49–88.

[33] L. Boltzmann, Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten, Wien. Ber. 58 (1868) 517–560.

[34] L. Boltzmann, Über das Wärmegleichgewicht der lebendigen Kraft zwischen mehratomigen Gasmolekülen, Wien. Ber. 63 (1871) 397–418.

[35] L. Boltzmann, Einige allgemeine Sätze über Wärmegleichgewicht, Wien. Ber. 63 (1871) 679–711.

[36] R. C. Tolman, The Principles of Statistical Mechanics, Oxford University Press, Oxford, 1938.

[37] D. ter Haar, Elements of Statistical Mechanics, Rinehart & Company, New York, 1954.

[38] D. Costantini, U. Garibaldi, A probabilistic foundation of elementary particle statistics. Part I, Stud. Hist. Phil. Mod. Phys. 28 (1997) 483–506.

[39] D. Costantini, U. Garibaldi, A probabilistic foundation of elementary particle statistics. Part II, Stud. Hist. Phil. Mod. Phys. 29 (1998) 37–59.

[40] L. Brillouin, Comparison des différent statistiques appliquées aux problèmes des quanta, Ann. Phys. Paris 7 (1927) 315–331.

[41] O. Penrose, Foundations of Statistical Mechanics, Hardcover Edition, Pergamon Press, Oxford, 1970.

[42] E. Scalas, E. Martin, G. Germano, Ehrenfest urn revisited: Playing the game on a realistic fluid model, Phys. Rev. E 76 (2007) 011104.

[43] P. J. Kraulis, MolScript — A program to produce both detailed and schematic plots of protein structures, J. Appl. Crystallogr. 24 (1991) 946–950, http://www.avatar.se/molscript (accessed Oct 13, 2010).

[44] W. Humphrey, A. Dalke, K. Schulten, VMD — Visual Molecular Dynamics, J. Mol. Graph. 14 (1996) 33–38, http://www.ks.uiuc.edu/research/vmd (accessed Oct 13, 2010).

[45] E. A. Merritt, D. J. Bacon, Raster3d: Photorealistic molecular graphics, Method. Enzymol. 277 (1997) 505–524, http://skuld.bmsc.washington.edu/raster3d (accessed Oct 13, 2010).

[46] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, T. E. Ferrin, UCSF Chimera — A visualization system for exploratory research and analysis, J. Comput. Chem. 25 (2004) 1605–1612, http://www.cgl.ucsf.edu/chimera (accessed Oct 13, 2010).

[47] J. Li, Atomeye: An efficient atomistic configuration viewer, Model. Simul. Mater. Sc. 11 (2003) 173–177.

[48] RasMol homepage, http://www.umass.edu/microbio/rasmol (accessed Oct 13, 2010).

[49] gopenmol homepage, http://www.csc.fi/gopenmol (accessed Oct 13, 2010).

[50] Jmol: an open-source java viewer for chemical structures in 3d, http://www.jmol.org (accessed Oct 13, 2010).

[51] W. L. Delano, The PyMOL molecular graphics system, DeLano Scientific, Palo Alto, CA, USA, http://pymol.sourceforge.net (accessed Oct 13, 2010).

[52] Molekel homepage, http://www.cscs.ch/molekel (accessed Oct 13, 2010).

[53] Discovery Studio homepage, http://accelrys.com/products/discovery-studio/ (accessed Oct 13, 2010).

[54] SYBYL homepage, http://www.tripos.com (accessed Oct 13, 2010).

[55] MOLCAD homepage, http://www.molcad.com (accessed Oct 13, 2010).

[56] G. Cinacchi, J. S. van Duijneveldt, Phase behaviour of contact lens-like particles: Entropy-driven competition between isotropic-nematic pase separation and clustering, J. Phys. Chem. Lett. 1 (2010) 787–791.

[57] J. G. Gay, B. J. Berne, Modification of the overlap potential to mimic a linear site-site potential, J. Chem. Phys. 74 (1981) 3316–3319.

[58] M. A. Bates, G. R. Luckhurst, Computer simulation studies of anisotropic systems. XXVI. Monte Carlo investigations of a Gay–Berne discotic at constant pressure, J. Chem. Phys. 104 (1996) 6696–6709.

[59] M. P. Allen, G. Germano, Expressions for forces and torques in molecular simulations using rigid bodies, Mol. Phys. 104 (2006) 3225–3235.

[60] T. Kihara, Convex molecules in gaseous and crystalline states, Adv. Chem. Phys. 5 (1963) 147–188.

[61] R. Berardi, C. Fava, C. Zannoni, A Gay-Berne potential for dissimilar biaxial particles, Chem. Phys. Lett. 297 (1998) 8–14.

[62] M. P. Allen, G. T. Evans, D. Frenkel, B. M. Mulder, Hard convex body fluids, Adv. Chem. Phys. 86 (1993) 1–166.

[63] B. Martinez-Haya, A. Cuetos, S. Lago, L. F. Rull, A novel orientation-dependent potential model for prolate mesogens, J. Chem. Phys. 122 (2005) 024908.

[64] C. Zannoni, Molecular design and computer simulations of novel mesophases, J. Mater. Chem. 11 (2001) 2637–2646.

[65] N. Attig, K. Binder, H. Grubmüller, K. Kremer (Eds.), Computational Soft Matter: From Synthetic Polymers to Proteins, Forschungszentrum Jülich, Jülich, 2004, http://www.fz-juelich.de/nic-series/volume23 (accessed Oct 13, 2010).

[66] M. R. Wilson, Progress in computer simulations of liquid crystals, Int. Rev. Phys. Chem. 24 (2005) 421–455.

[67] C. Amovilli, I. Cacelli, G. Cinacchi, L. D. Gaetani, G. Prampolini, A. Tani, Structure and dynamics of mesogens using intermolecular potentials derived from ab initio calculations, Theor. Chem. Acc. 117 (2007) 885–901.

[68] G. Prampolini, Parametrization and validation of coarse grained force-fields derived from ab initio calculations, J. Chem. Theory Comput. 2 (2006) 556–567.

[69] A. Liwo, M. Khalili, C. Czaplewski, S. Kalinowski, S. Ołdziej, K. Wachucik, H. A. Scheraga, Modification and optimization of the united-residue (UNRES) potential energy function for canonical simulations. I. Temperature dependence of the effective energy function and tests of the optimization method with single training proteins, J. Phys. Chem. B 111 (2007) 260–285.

[70] J. S. Richardson, The anatomy and taxonomy of protein structure, Adv. Protein Chem. 34 (1981) 167–339.

[71] J. S. Richardson, Schematic drawings of protein structures, Method. Enzymol. 115 (1985) 359–380.

[72] B. Lee, F. M. Richards, The interpretation of protein structures: Estimation of static accessibility, J. Mol. Biol. 55 (1971) 379–400.

[73] M. L. Connolly, Analytical molecular surface calculation, J. Appl. Crystallogr. 15 (1983) 548–558.

[74] M. L. Connolly, Solvent-accessible surfaces of proteins and nucleic acids, Science 221 (1983) 709–713.

[75] M. L. Connolly, The molecular surface package, J. Mol. Graph. 11 (1993) 139–141.

[76] C. Chiccoli, P. Pasini, F. Semeria, C. Zannoni, Three-dimensional visualization of molecular organization and phase transitions in liquid crystal lattice models, Int. J. Mod. Phys. C 3 (1992) 1209–1220.

[77] R. Berardi, A. P. J. Emerson, C. Zannoni, Monte Carlo investigations of a Gay-Berne liquid crystal, J. Chem. Soc. Faraday T. 89 (1993) 4069–4078.

[78] Avs/express homepage, http://www.avs.com (accessed Oct 13, 2010).

[79] POV-Ray homepage, http://www.povray.org (accessed Oct 13, 2010).

[80] OpenGL homepage, http://www.opengl.org (accessed Oct 13, 2010).

[81] A. T. Gabriel, T. Meyer, G. Germano, Molecular graphics of convex body fluids, J. Chem. Theory Comput. 4 (2008) 468–476.

[82] QMGA homepage, http://qmga.sourceforge.net (accessed Oct 13, 2010).

[83] D. Caprion, L. Bellier-Castella, J.-P. Ryckaert, Influence of shape and energy anisotropies on the phase diagram of discotic molecules, Phys. Rev. E 67 (2003) 041703.

[84] R. B. Corey, L. C. Pauling, Molecular models of amino acids, peptides, and proteins, Rev. Sci. Instrum. 24 (1953) 621–627.

[85] W. L. Koltun, Precision space-filling atomic models, Biopolymers 3 (1965) 665–679.

[86] C. Stillings, E. Martin, M. Steinhart, R. Pettau, J. Paraknowitsch, M. Geuss, J. Schmidt, G. Germano, H. W. Schmidt, U. Gösele, J. H. Wendorff, Nanoscaled discotic liquid crystal/polymer systems: Confinement effects on morphology and thermodynamics, Mol. Cryst. Liq. Cryst. 495 (2008) 285.

[87] Ffmpeg homepage, http://ffmpeg.org (accessed Oct 13, 2010).

[88] Avidemux homepage, http://avidemux.sourceforge.net (accessed Oct 13, 2010).

[89] Qt homepage, http://www.trolltech.com/products/qt (accessed Oct 13, 2010).

[90] VRML97 and related specifications, http://www.web3d.org/x3d/specifications/vrml (accessed Oct 13, 2010).

[91] X3d and related specifications, http://www.web3d.org/x3d (accessed Oct 13, 2010).

[92] M. R. Wilson, M. P. Allen, M. A. Warren, A. Sauron, W. Smith, Replicated data and domain decomposition molecular dynamics techniques for simulation of anisotropic potentials, J. Comput. Chem. 18 (1997) 478–488.

[93] Coin3d homepage, http://www.coin3d.org (accessed Oct 13, 2010).

[94] OpenInventor homepage, http://oss.sgi.com/projects/inventor (accessed Oct 13, 2010).

[95] H. Goldstein, C. P. Poole, J. L. Safko, Classical Mechanics, 3rd Edition, Addison Wesley, San Francisco, 2002.

[96] C. Giberti, L. Rondoni, C. Vernia, Asymmetric fluctuation-relaxation paths in FPU models, Physica A 365 (2006) 229–234.

[97] C. P. Searles, D. J. Searles, L. Rondoni, Temporal asymmetry of fluctuations in nonequilibrium steady states., J. Chem. Phys. 124 (2006) 114109.

[98] C. Giberti, L. Rondoni, C. Vernia, Temporal asymmetry of fluctuations in the nonequilibrium FPU model, Physica D 228 (2007) 64–76.

[99] C. Paneni, D. J. Searles, L. Rondoni, Temporal asymmetry of fluctuations in nonequilibrium steady states: Links with correlation functions and nonlinear response, J. Chem. Phys. 128 (2008) 164515.

[100] D. L. Cheung, Monte Carlo simulations of liquid crystals between microstructured substrates, J. Chem. Phys. 128 (2008) 194902.

[101] Z. W. Li, L. J. Chen, Y. Zhao, Z. Y. Lu, Ordered packing of soft discoidal system, J. Phys. Chem B 112 (2008) 13842–13848.

[102] D. L. Cheung, S. A. F. Bon, Interaction of nanoparticles with ideal liquid-liquid interfaces, Phys. Rev. Lett. 102 (2009) 066103.

[103] J. S. Lintuvuori, M. R. Wilson, A coarse-grained simulation study of mesophase formation in a series of rod-coil multiblock copolymers, Phys. Chem. Chem. Phys 11 (2009) 2116–2125.

[104] Z. W. Li, Z. Y. Sun, Z. Y. Lu, Simulation model for hierarchical self-assembly of soft disklike particles, J. Phys. Chem B 114 (2010) 2353–2358.

[105] G. Cinacchi, J. S. van Duijneveldt, Phase behavior of contact lens-like particles: Entropy-driven competition between isotropic-nematic phase separation and clustering, J. Phys. Chem. Lett. 1 (2010) 787–791.

[106] J. S. Lintuvuori, M. R. Wilson, Statistical temperature molecular dynamics simulations applied to phase transitions in liquid crystalline systems, J. Chem. Phys. 132 (2010) 224902.

[107] C. K. Lee, C. C. Hua, S. A. Chen, Parametrization of the Gay-Berne potential for conjugated oligomer with a high aspect ratio, J. Chem. Phys. 133 (2010) 064902.

[108] E. Abrahamsson, S. S. Plotkin, BioVEC: A program for biomolecule visualization with ellipsoidal coarse-graining, J. Mol. Graph. Model. 28 (2009) 140–145.

[109] Cygwin homepage, http://www.cygwin.com/ (accessed Oct 13, 2010).

Danksagung

Ich möchte an dieser Stelle meinem Doktorvater Professor Guido Germano für die Unterstützung im Rahmen dieser Dissertation danken.
Ebenfalls möchte ich mich bei meinen Kollegen Dr. Edgar Martin, Dr. Mauro Politi, Dr. Daniel Fulger und Ulrich Welling für das angenehme Arbeitsklima bedanken.
Timm Meyer danke ich für die Unterstützung bei dem OpenGL-Teil von QMGA.
Professor Michael Guthe danke ich für das professionelle Auge, welches er auf QMGA geworfen hat, und für die Grundsteinlegung der VBOs.
Professor Enrico Scalas danke ich für das Aufwerfen der Fragestellung zu den Verteilungsfunktionen bei Gasen von harten Kugeln mit endlichen Teilchenzahlen und für seinen Beitrag zur Entwicklung der dazugehörigen Theorie.
Markus P. Schmidt danke ich für die Grundsteinlegung des Integrators des Lorentzgasprogramms.
Ich bedanke mich auch bei den Studenten meiner Übungsgruppen. Es hat mir immer Freude bereitet, Euch Mathe einzuhämmern. ;-)
Privat möchte ich mich bei meiner Freundin Rebecca Sure für ihre Geduld und Unterstützung in den Aufs und Abs während meiner Dissertation bedanken.
Und natürlich möchte ich mich auch bei meinen Eltern bedanken; sie haben mich in allen meinen Entscheidungen immer unterstützt.

Ich habe diese Arbeit selbstständig erstellt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Adrian Gabriel, Marburg, Oktober 2010